Depth estimation for monocular Direct Visual Odometry

Ran Cheng

Master of Science

School of Computer Science
McGill University
Montreal, Quebec, Canada

May 2020

A thesis submitted to McGill University in partial fulfillment of the requirements of the degree of Master of Science

Abstract

Direct monocular visual odometry can be computed using an algorithm for estimating camera motion by directly comparing image brightness on a monocular camera. Its excellent performance and stability make it very popular in the field of visual navigation. Under appropriate initialization conditions, direct image comparison allows the direct method to perform relative positioning without a map, which enables the algorithm to work in many environments without external positioning equipment, such as indoor and underground. However, because the direct method is based on direct comparison of image pixels, and the direct image comparison loss function is non-convex, makes it difficult to use this method to solve for the pose. In addition, the lack of depth information in the image also causes scale drift in the localization process of the monocular direct method. In previous work, an indirect (corner features) component has been introduced to correct tracking scales with offline loop closures. We further propose and examine the use of end-to-end deep CNN components to predict the image depth and camera poses in a self-supervised fashion. We exploit depth prediction as a candidate prior for the coarse initialization, tracking, and marginalization steps of the direct visual odometry system, enabling the second-order optimizer to converge faster to a precise global minimum. In addition, the given depth prior supports large baseline stereo scenarios, maintaining robust pose estimation against challenging motions such as in-place rotation. We further refine our pose estimation with semi-online loop closure. The experiments on KITTI demonstrate that our proposed method achieves state-of-the-art performance compared to both traditional direct visual odometry and learning-based counterparts.

Abrégé

L'odométrie visuelle monoculaire directe peut être calculée à l'aide d'un algorithme pour estimer le mouvement de la caméra en comparant directement la luminosité de l'image sur une caméra monoculaire. Ses excellentes performances et sa stabilité le rendent très populaire dans le domaine de la navigation visuelle. Dans des conditions d'initialisation appropriées, la comparaison directe d'images permet à la méthode directe d'effectuer un positionnement relatif sans carte, ce qui permet à l'algorithme de fonctionner dans de nombreux environnements sans équipement de positionnement externe, comme intérieur et souterrain. Cependant, étant donné que la méthode directe est basée sur une comparaison directe des pixels de l'image et que la fonction de perte de comparaison d'image directe n'est pas convexe, il est difficile d'utiliser cette méthode pour résoudre la pose. De plus, le manque d'informations de profondeur dans l'image provoque également une dérive d'échelle dans le processus de localisation de la méthode monoculaire directe. Dans les travaux précédents, un composant indirect (caractéristiques des coins) a été introduit pour corriger les échelles de suivi avec des fermetures de boucle hors ligne. Nous proposons et examinons en outre l'utilisation de composants CNN profonds de bout en bout pour prédire la profondeur de l'image et les poses de la caméra de manière auto-supervisée. Nous exploitons la prédiction de la profondeur en tant que candidat avant les étapes d'initialisation grossière, de suivi et de marginalisation du système d'odométrie visuelle directe, permettant à l'optimiseur du second ordre de converger plus rapidement vers un minimum global précis. De plus, la profondeur donnée prend en charge de grands scénarios stéréo de base, conservant une estimation de pose robuste contre les mouvements difficiles tels que la rotation en place. Nous affinons davantage notre estimation de pose avec une fermeture de boucle semi-en ligne. Les expériences sur KITTI démontrent que la méthode que nous proposons offre des performances de pointe par rapport à l'odométrie visuelle directe traditionnelle et à ses homologues basés sur l'apprentissage.

Contributions

This thesis was initially inspired and influenced by the work of Manderson *et al.* [MHCD18] [MCMD18], where I collaborated and contributed to the visual odometry part. In these two publications, we have explored the potential of visual navigation under challenging environments (e.g., underwater). I have customized an existing direct odometry method from other authors, Direct Sparse Odometry [EKC17] for the underwater exposure affine model. I have also proposed a real-time tracking quality metric. This metric helps robots to decide which region is bad for visual navigation so that they can avoid collisions. These experiences helped me to build expertise in direct visual odometry methods. They also offered me enough experience to deeply integrate deep learning methods with traditional visual odometry and finally resulted in the development of this thesis.

The major part of this thesis, including method and experiments, is from Cheng *et al.* [CAMD20], where I am the major contributor. I proposed the approach, implemented the methods, and wrote the paper. Collaborator Christopher Agia did the experiments on runtime efficiency and refined the writing. My supervisors David Meger and Gregory Dudek, helped me editing the paper and gave me constructional advice. I further extended the method with a gradient-based residual pattern and the sparse to dense depth prediction modules.

Acknowledgements

Thank you to my supervisors, Professor Gregory Dudek and David Meger, for providing guidance and feedback throughout the project.

Contents

List of Figures

List of Tables

1 Introduction

At the beginning of the establishment of computer vision, people imagined that computers would one day observe the world through their eyes, understand the surrounding objects, and explore the field of location [HZ03]. This is a wonderful and romantic dream that has attracted countless scientific researchers. However, progress is far from what was expected. The mountains, grass, trees, fish, and beasts are captured by our eyes in such a different fashion than used by computers. They are just matrices of numbers. Enabling a computer to understand the content of images is as difficult as letting us understand these numbers. We were puzzled for a long time, and until today, only a few signs of success have been found. Through deep learning, computers can recognize objects, faces, sounds, and words. At the same time, after the development of SLAM (Simultaneous Localization and Mapping) for nearly 30 years [LH81], some researchers can now use images to track their position in real-time, and some can even perform the real-time 3D reconstruction [DRMS07].

What is more exciting is that in recent years, with the development of visual SLAM, many related applications have emerged: indoor sweeping robots, unmanned vehicles in the wild, drones in the air, and virtual and augmented reality. They all need to know where they are. Visual positioning is so important. Without it, the sweeping robot cannot plan its own path, the home robot cannot send breakfast to our bedside according to the instructions, and the virtual reality will always be fixed on the seat. None of these interesting things can appear in real life. How pitiful that would be.

Since the 21st century, the continuous breakthrough of visual SLAM in theory and practice has made it gradually walk out of the laboratory to industry, and gradually form the following common parts:

- Sensory processing is responsible for image acquisition and preprocessing. If it is in a robot, it also needs to read and synchronize information such as wheel odometer and inertial sensor.
- Visual odometry estimates the camera movement between adjacent images and builds a local map, which has become the front end of visual SLAM.
- Backend optimization takes the camera posture measured by the visual odometer at different times, and the loop closing detection information. It optimizes them uniformly to obtain a globally consistent track and map. Because it is processed immediately after visual odometry, it is also called backend.
- Loop closing determines whether the robot has reached the previous position. If a loop closing is detected, it will provide the information to the backend for processing. This part is usually run under a separate thread.
- Mapping builds a corresponding 3D map based on the estimated and optimized camera trajectory.

Before camera technology matured, the frame rate was not high enough, the imaging was mainly rolling shutter, and the noise caused by unstable exposure often had a great impact on the camera motion estimation of the visual odometry. As a result, the visual odometry was not so reliable, and often required continuous optimization and correction at the back end to prevent accumulated errors. However, with the improvement of imaging technology, the high frame rate, and smooth exposure have improved the accuracy of the visual odometer's estimation of the relative motion of the camera. Under good initialization conditions, the visual odometer can even achieve the same level of accuracy as the optimized one. In the following content, we will introduce the current mainstream visual odometry methods and how they work. At the end of this chapter, we discuss how to further optimize the visual odometry methods through deep learning.

1.1 Visual odometry brief introduction

Visual odometry methods [FPRARM15] can be divided into two major categories: indirect visual odometry methods and direct visual odometry methods [PKK18]. Indirect visual odometry methods extract the sparse feature points from each image frame and estimate relative motion by matching the features. While feature-based indirect visual odometry methods only use sparse information from images, direct visual odometry methods can use all pixels in the image to build the dense map and deliver more robust localization performance with photometric consistency. Unlike indirect visual odometry methods that build the correspondence between features to solve for relative camera poses, direct image alignment usually needs a very good initial guess on the pixel-wise depth and relative pose of two consecutive image frames, because direct methods have no data association or indexing process.

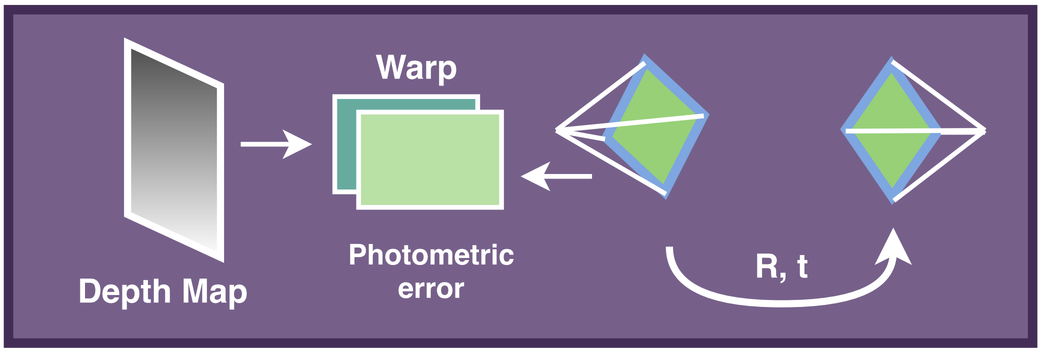


Figure 1.1: Joint optimization of bundle adjustment.

Depth estimation and pose estimation are two fundamental components of the direct visual odometry method [MLD+09] and are usually optimized together. Since the optimization adjusts two ends of the light ray, people also name this optimization process Bundle Adjustment. As shown in Fig 1.1, by adjusting the depth and relative camera poses, pixels in one frame can be warped into another one and generate a distance between either extracted features or local patch intensity distributions. Bundle adjustment jointly optimizes the depth and camera pose together to minimize the total pixel-wise warping distance, namely reprojection error or photometric error. To minimize those errors, feature based (indirect) visual odometry methods use RANSAC algorithm and semi-local constraints on scale/rotation invariant nearest neighbor descriptors; whereas direct methods exploit brightness consistency and robust photometric error on image transformations to solve the problem.

Direct visual odometry methods [FPS14][ESC14][EKC17] have received more attention in recent years after the popularization of global shutter and high resolution cameras. Direct image alignment, such as optical flow and its variants, optimize a photometric error (difference of image pixel intensity) to estimate camera ego-motion and are well known for their speed and precision. Feature extraction from images is not required for such methods, and thus, they are able to work in feature deficient environments (even on pure gradient images). However, because the image intensity is not smooth in both global and local regions, it is difficult to find the optimal solution smoothly by the gradient optimization algorithm that directly compares the image intensity (as shown in Fig. 1.3). Moreover, photometric consistency is a very strong assumption that is usually not guaranteed in practice, such as when entering a tunnel or being covered by the shadow of a tree.

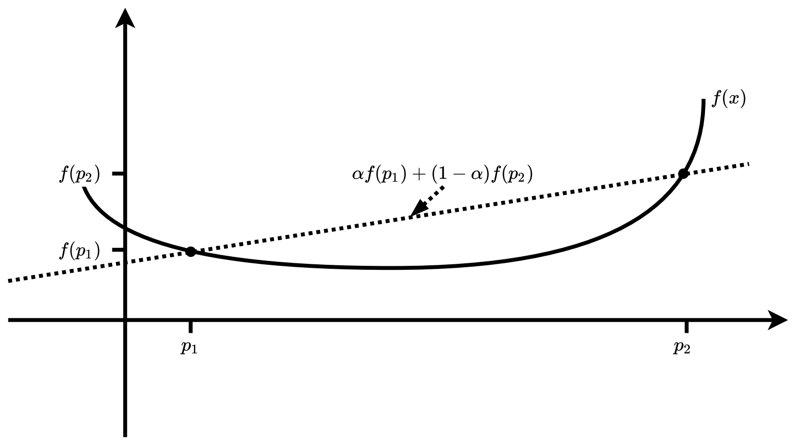


Figure 1.2: A convex function

In order not to be confused with deep learning or convex optimization in optimization theory, we simply define the convex nature of the image here. As shown in Fig. 1.3, the image intensity manifold in the 2D surface is not a smooth and may have many local minima.

Let I be the grayscale image intensity matrix ($H \times W \times 1$), and let $f : I \rightarrow R$ be the image intensity function that can return the intensity given a pair of pixel coordinate $p = [u, v]$. We assume two points p_1 and p_2 are two arbitrary pixel points in the image I.

f is called convex if:

$$\forall p_1, p_2 \in I, \forall t \in [p_1, p_2] : f(\alpha p_1 + (1 - \alpha)p_2) \leqslant \alpha f(p_1) + (1 - \alpha)f(p_2) \tag{1.1}$$

Clearly most of the point pairs in Fig. 1.3 do not meet the above conditions. Thus we call the image f a non-convex manifold. The non-convex manifold will be passed into the error manifold when direct visual odometry tries to compare two images directly. This non-convexity is very difficult to optimize, especially for the algorithms relying on gradient such as gradient descend and Gauss-Newton method.

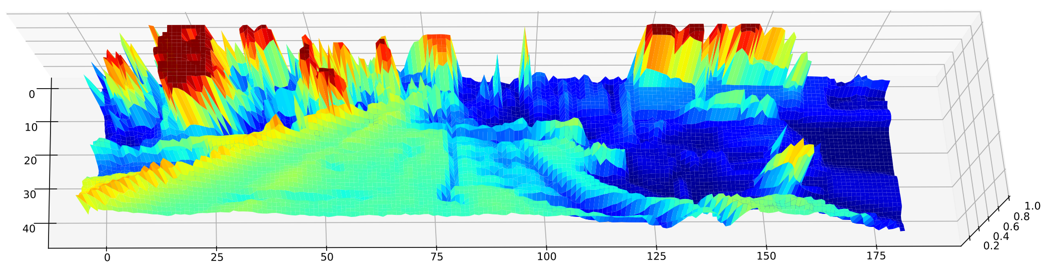
At present, several prior authors have developed methods to circumvent the impact of this non-convex problem formulation in visual navigation. For example, increasing the camera frame rate makes the transformation between the two images small, so that we can find convex areas in the local range to guide optimization. At the same time, researchers are also looking for solutions through deep neural networks.

1.2 Deep learning for visual odometry

Many models have been proposed to increase the efficiency and robustness of direct visual odometry methods. Illumination-robust cost [WP19] and rigid body mutual information [SM16] have been proposed to compensate for drastic illumination changes between frames. The photometric calibration model [BWC17] has been employed as a service to estimate camera exposure time, gamma reaction, and vignette to adapt for auto-exposure cameras. Further, the adoption of reprojection residual patterns have been shown to smoothen the photometric error manifold, increasing the rate of convergence for gradient-based optimization.

Deep learning based methods have demonstrated remarkable progress on maintaining consistent scale for both depth prediction and camera pose estimation. It is noteworthy to mention that the majority of deep learning models are actually predicting inverse depths (disparity maps). With proper scaling, we can seamlessly feed this inverse depth prior into

(a) RGB image


(b) intensity manifold

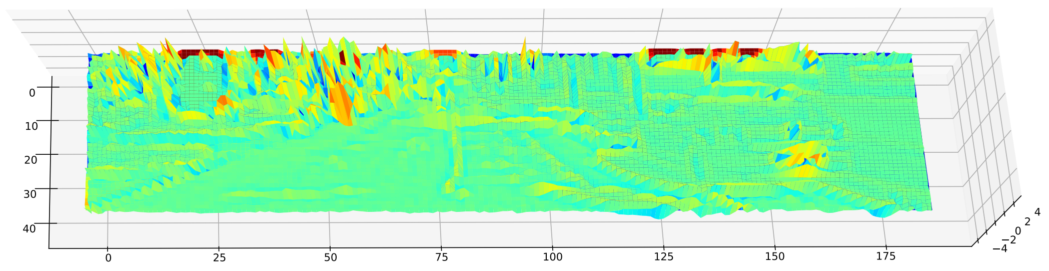
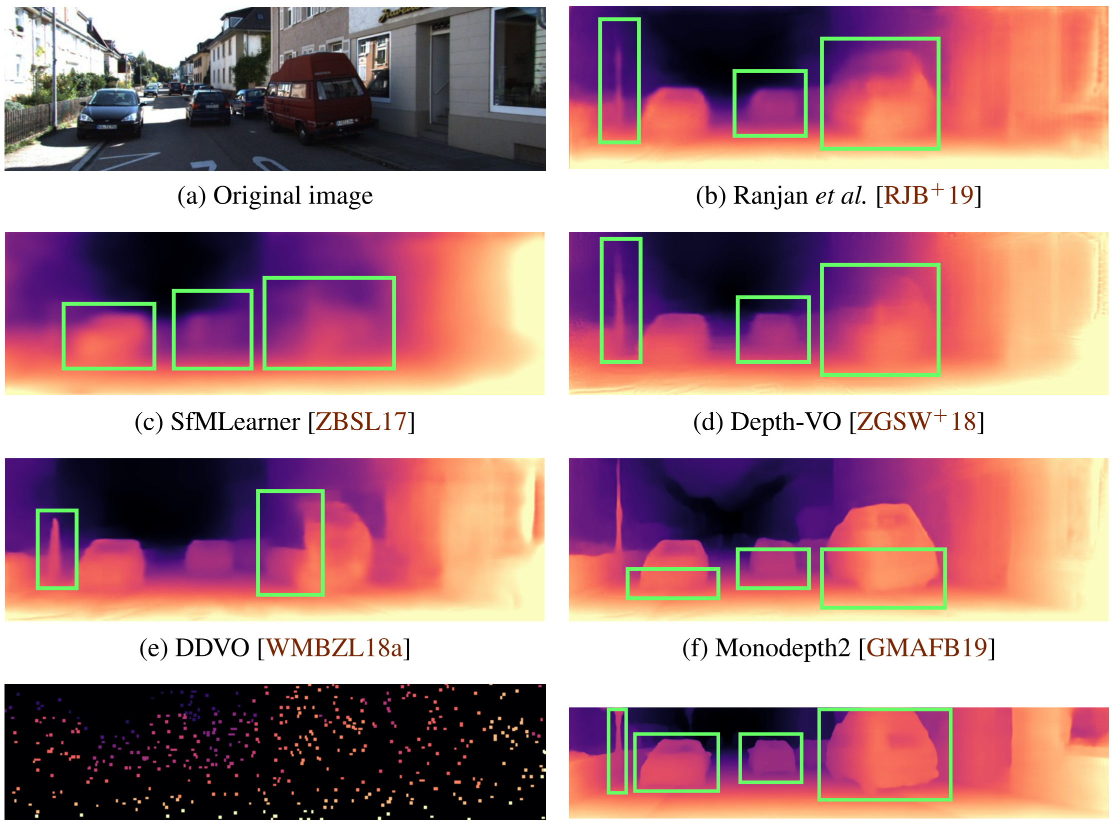

(c) gradient manifold

Figure 1.3: Noisy image intensity and its very bumpy gradient manifold.

the depth filter module in the direct visual odometry system. This addition can aid in reducing the effect of scale-drift for successive tracking and mapping tasks.

Some notable deep learning methods including DeepVO [WCWT17], SfM Learner [ZBSL17], DeepSfM [WZL$^+$19] and BANet [TT18] which together achieved remarkable progress on both dense depth estimation and camera pose estimation. Most depth prediction methods share the same ill-posed problem formulation: there are a large number of possible incorrect depths per pixel, which can be considered *perfect matches* with respect to photometric loss. The noisy error manifold resulting from this ambiguity leads to high-variance error gradient being computed (a key element in learning tasks) so that the com-



(a) Original image (b) Ranjan *et al.* [RJB+19]

(c) SfMLearner [ZBSL17] (d) Depth-VO [ZGSW+18]

(e) DDVO [WMBZL18a] (f) Monodepth2 [GMAFB19]

(g) Ours, sparse map from tracked points (h) Ours, sparse-to-dense map

Figure 1.4: Depth prediction from monocular image.

munity introduced the smoothness loss [GMAB17] [ZGSW+18] to wipe out those outliers (low-gradient local regions) in disparity. The introduction of smoothness loss regularized the overall prediction yet can lead to blurry on object boundaries which are exactly the high gradient regions. This is the reason why most of the depth predictions are blurry, and there are remarkably large prediction errors observed on most object boundaries, as shown in Fig. 1.4. Note that in Fig. 1.4, the prediction of our dense depth map is restored based on the map points that we successfully tracked in the running visual odometer. Hence, the uniformity and accuracy of the depth prediction are better than that obtained from deep learning inference on a single frame image. Among the sequences of KITTI odometry dataset, our depth estimation demonstrates better performance, embodied in clear object edges and smooth wall surfaces. Detailed evaluation results can be found in the Experiments chapter.

1.3 Our proposal and contributions

Our proposal: In this work, we present a real-time monocular visual odometry system with an auxiliary deep depth predictor. In visual navigation algorithms, depth inference and relative camera pose estimation are usually performed simultaneously, but we can simplify the above algorithm into a process of seeking the marginal distribution when any variable is known. With the depth information predicted from a deep CNN network, the joint optimization can thus be broken down into two stages: **a)** solve the pose from a coarse depth prior; **b)** correct the depth estimate from the pose given by **a**. Unlike Zhao *et al.* [ZTS19] and Loo *et al.* [LAM$^+$19] that only use depth for direct image alignment or to regularize the depth filter, we also incorporate the depth priors into the tracking and marginalization backend. We argue that the use of a deep network predicted pose to initialize direct image alignment is unnecessary since the decoupled pose can be recovered from a coarse depth prior efficiently. In the ablation study described later in this thesis, the pose network initialization stage is shown to provide only minor improvements to our final results. Inspired by LDSO [GWDC18], we divide the direct visual odometry problem into two main parts: coarse tracking and map refinement. Instead of performing global photometric bundle adjustment, we select keyframes based on marginalization results and form a pose graph by encoding ORB (Oriented FAST and rotated BRIEF) features using a BoW (Bag of Words) vector quantization approach. These corner features are uniformly selected from the image space to ensure that the mapped points can be reproduced in a later frame for proper loop closure.

Further, we propagate the converged sparse depth map in visual odometry backend to refine the global scales of dense depth inferences. We achieve state-of-the-art pose estimation performance in the KITTI dataset, as shown in Fig. 1.5. Note that LDSO [GWDC18] is not running in real-time like ours. Also, our scale corrected depth results improved the disparity background estimation for most metrics, including absolute relative error, squared relative error, root mean squared error and log of root mean squared error.

Our contributions: First, our model expedites initialization in direct visual odometry method with consistent tracking scale: Given the disparity map and baseline factor, we can easily recover the depth map from the depth estimator. However, current depth estima-

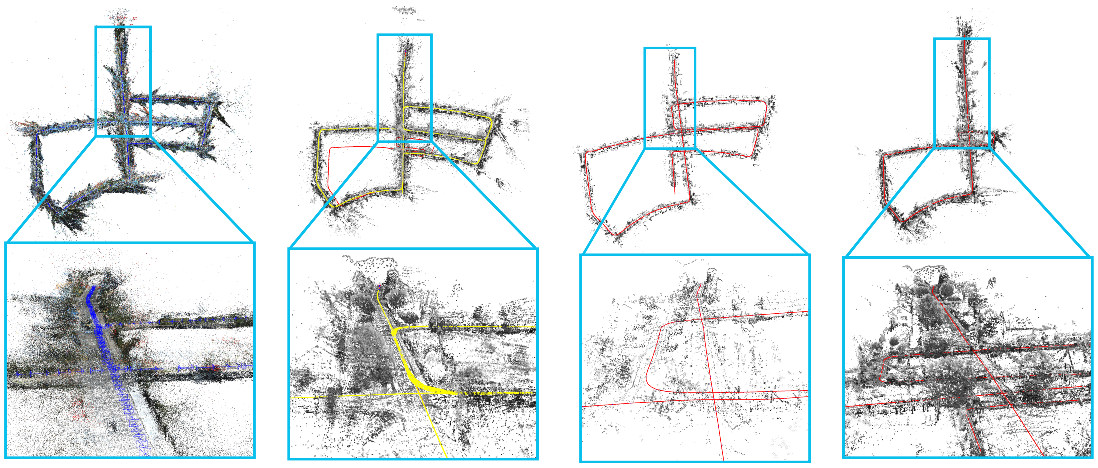

(a) Ours (b) LDSO [GWDC18] (c) SDSO [WSC17] (d) DSO [EKC17]

Figure 1.5: Example mapping results on KITTI seq 05.

tors are coarse and error-prone at the boundaries due to their smoothness loss. Similarly, initialization in traditional VO is extremely time-consuming and usually converges to arbitrary scale-spaces. Using the depth prediction from a deep network accelerates the VO initialization and keeps a consistent scale as the tracking goes on. Second, the depth filter in the direct visual odometry method refines the depth estimation for each tracking frame, which explores a novel possible method to learn the depth in self-supervised architectures.

Second, we leverage the dense depth map in loop closing and marginalization steps when the pose and map points are updated upon the tracking window. With the dense depth prior, our depth filter can converge reasonably fast, which compensates the time cost of depth inference. Although the depth prior is noisy, if may offer the optimizer a locally convex (monotone) initial point from which to start descent. This initial prior improves our pose estimation accuracy and robustness within large scale in-place rotation scenarios.

Finally, we carefully model the influence of non-convexity introduced in the direct visual odometry method energy function and propose a novel self-supervised learning model to isolate the image from the residual-energy function and removed the image gradient from the pose optimization loop to secure the robustness of gradient-based tracking backend. This proposed pipeline can further help the direct visual odometry method to expand into more challenging video processing use cases.

1.4 Organization of this thesis

This thesis introduces the underlying theories that underlay direct visual odometry method and the design pattern of DSO in Chapter 2. It then discusses related works in Chapter 3 and introduces our novel method in Chapter 4 which introduces how to integrate the depth estimated from deep CNN in the initialization and tracking process. Chapter 5 demonstrates the evaluation results on the KITTI dataset and we finally conclude the thesis in Chapter 6.

2 Background

In this section, we present the underlying theory of direct visual odometry. By understanding how depth and camera pose are estimated, we can find the proper way to integrate the deep CNN into the direct visual odometry pipeline.

2.1 Optical flow

Unlike the indirect method, which usually solves the camera pose and depth by feature extraction and data association, the direct method adopts the essence of optical flow [HS81] (a fundamental method of calculating the motion of image intensities), which directly compares the pixel intensity to derive the camera pose. By avoiding feature extraction, direct methods are robust in the low-feature environment, as long as the gradient of the image exists [SMR08].

As direct methods come from optical flow, to better illustrate the direct visual odometry, we will first introduce a common optical flow method. Optical flow assumes the pixels from the same 3D location maintain constant brightness [Gib02]. Let's assume that a pixel located at (x, y) at time t moves to $(x+d_x, y+d_y)$ at time $t+d_t$.Then we have the following equation:

$$I(x + d_x, y + d_y, t + d_t) = I(x, y, t). \tag{2.1}$$

Note that this brightness constancy assumption [IA99] is actually very strong in terms of the way materials reflect the light and the exposure configuration of cameras. Although we cannot guarantee the brightness consistency assumption of holds on particular candidate, at least we can improve the stability of tracking by sampling multiple candidates on the entire image many tracking candidates and benefit from quantity. Let's assume we are tracking on a candidate that holds the assumption above, and we can expand Equation (2.1) with Taylor approximation:

$$I(x+d_x, y+d_y, t+d_t) \approx I(x,y,t) + \frac{\partial I}{\partial x}d_x + \frac{\partial I}{\partial y}d_y + \frac{\partial I}{\partial t}d_t. \tag{2.2}$$

As we have constant brightness assumption, the pixel intensity should be equal on frame at $t + d_t$. Thus the expansion terms will be zero as following:

$$\frac{\partial I}{\partial x}d_x + \frac{\partial I}{\partial y}d_y + \frac{\partial I}{\partial t}d_t = 0. \tag{2.3}$$

Dividing both side by d_t will result in:

$$\frac{\partial I}{\partial x}\frac{d_x}{d_t} + \frac{\partial I}{\partial y}\frac{d_y}{d_t} = -\frac{\partial I}{\partial t} \tag{2.4}$$

where d_x/d_t is the velocity along the x axis in image space, and d_y/d_t the y axis velocity. We can denote the two velocities as $[u, v]^T$. $\partial I/\partial x$ and $\partial I/\partial y$ are image gradients along the horizontal and vertical directions. In the 2D case, our goal is to estimate the motion variables: u, v, which requires more than two equations (tracking points). Although the motion states in 3D cases are extended to 6 degrees of freedom and the derivation will be slightly different, it shares the same derivative property and brightness constancy assumption. We have not provided the full 6 degrees of freedom derivation here, but please refer Optical Flow Estimation [Jep] for further details.

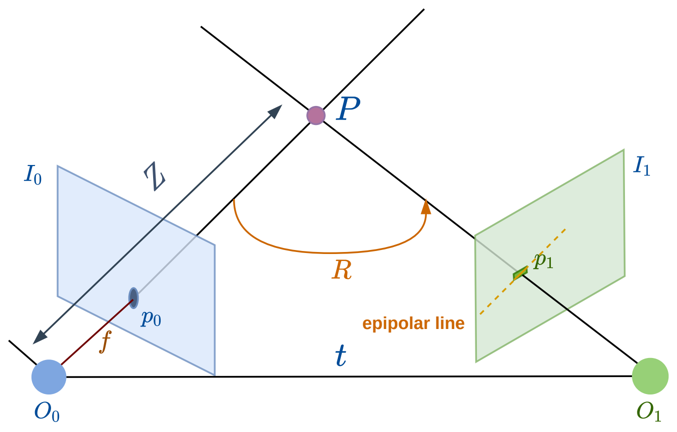


Figure 2.1: Triangulation illustration.

2.2 Derivation of the direct method

Weickert *et al.* [WBBP06] and Gao *et al.* [GZLY17] discussed the derivation process of optical flow method and direct method in detail. This paper simplified some of the derivation formulas on their basis and showed intuitively how the direct method solved the camera pose by Gauss-Newton method.

We first introduce the principle of the direct method and then expand its implementation step by step in the subsequent paragraphs.

All the following formulas are derived in the Lie group by default. The Lie group refers to a group with continuous properties. Discrete groups like integer groups Z have no continuous nature, so they are not Lie groups [Ced13]. In this thesis we only care about two special Lie groups, a special orthogonal group ($SO(3)$) composed of a three-dimensional rotation matrix R and a special Euclidean group ($SE(3)$) composed of a transformation matrix. We denote $\xi \in SE(3)$ as our camera pose. The rotation matrix R is a 3 by 3 matrix which describes the rotation of the rigid body, and the translation vector t is a 3 by 1 vector which describes the translation of the rigid body.

As shown in Fig. 2.1, point P in world is observed in both I_0 and I_1. We denote the world coordinate of P as $[X, Y, Z]$ and its projected pixel coordinates in the two corresponding image frames as p_0 and p_1. Our objective is now finding the relative 3D transformation $[R, t]$, denote as ξ in Lie group from I_0 to I_1. Meanwhile, let's assume the camera intrinsic is K which contains focal length f and image center coordinates $[c_x, c_y]$. The projection equations are:

$$p_0 = \begin{bmatrix} u_0 \\ v_0 \\ 1 \end{bmatrix} = \frac{1}{Z_0} KP, \text{ and} \tag{2.5}$$

$$p_1 = \begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix} = \frac{1}{Z_1} K(RP + t) = \frac{1}{Z_1} K(\text{exp}(\xi^\wedge)P). \tag{2.6}$$

Here Z_0, Z_1 are the depth in I_0, I_1 respectively. Since Lie group operation exp$(\xi^\wedge)$ is under homogeneous coordinates, we only need to take the first 3 elements. Recall that we are assuming the constant brightness at the same tracking point in two frames, we have:

$$I_0(p_0) = I_1(p_1). \tag{2.7}$$

To find the best relative transformation matrix $[R, t]$ we need to find the local minimal of the intensity difference (also known as Photometric Error):

$$min_\xi J(\xi) = ||I_0(p_0) - I_1(p_1)||^2. \tag{2.8}$$

However, the constant brightness assumption is too strong for a single point. So, typically, we consider N points to guarantee robustness [FPS14]. Our estimation of camera pose, in turn, will become:

$$min_\xi J(\xi) = \sum_{i=1}^{N} e_i^T e_i, \quad e_i = I_0(p_{0,i}) - I_1(p_{1,i}). \tag{2.9}$$

Note that the variable to optimize is ξ. To solve the least square optimization problem, we need to figure out how the error change in terms of the camera pose ξ. By adding a small incremental term in $\exp(\xi)$ in Lie manifold [Mur62], the tangent space will remain the same as following:

$$\begin{aligned} e(\xi \oplus \delta\xi) &= I_0\left(\tfrac{1}{Z_0}KP\right) - I_1\left(\tfrac{1}{Z_1}K\text{exp}(\delta\xi^\wedge)\text{exp}(\xi^\wedge)P\right) && (2.10)\\ &= I_0\left(\tfrac{1}{Z_0}KP\right) - I_1\left(\tfrac{1}{Z_1}K\text{exp}(\delta\xi^\wedge)P + \tfrac{1}{Z_1}K\delta\xi^\wedge\text{exp}(\xi^\wedge)P\right). && (2.11)\end{aligned}$$

The second term in equation (2.10) can be regarded as first-order approximation of the projection function. For better illustration, we denote:

$$\begin{aligned} q &= \delta\xi^\wedge\text{exp}(\xi^\wedge)P, && (2.12)\\ u &= \frac{1}{Z_1}Kq, && (2.13)\end{aligned}$$

where q is the 3D coordinate in the second image frame, and u is the corresponding pixel coordinate. Equation (2.10) is thus equivalent to:

$$e(\xi \oplus \delta\xi) = e(\xi) - \frac{\partial I_1}{\partial u}\frac{\partial u}{\partial q}\frac{\partial q}{\partial \delta\xi}\delta\xi. \tag{2.14}$$

From the above equation, we can find that the Jacobian of $\delta\xi$ with respect to the photometric error in Lie manifold is:

$$J_{\delta\xi} = \frac{\partial I_1}{\partial u}\frac{\partial u}{\partial \delta\xi}. \tag{2.15}$$

Here J is the Jacobian of the image in terms of motion update $\delta\xi$. Since $\partial u/\partial q$ and $\partial q/\partial\delta\xi$ only depend on the geometric parameters (ξ, the relative pose and Z_1, the depth in second frame coordinate), we can treat them as a whole.

$$\frac{\partial u}{\partial \delta \xi} = \begin{bmatrix} \frac{f_x}{Z_1} & 0 & -\frac{f_x X_1}{Z_1^2} & -\frac{f_x X_1 Y_1}{Z_1^2} & f_x + \frac{f_x X_1^2}{Z_1^2} & -\frac{f_x Y_1}{Z_1^2} \\ 0 & \frac{f_y}{Z_1} & -\frac{f_y Y_1}{Z_1^2} & -f_y - \frac{f_y Y_1^2}{Z_1^2} & -\frac{f_y X_1 Y_1}{Z_1^2} & -\frac{f_y X_1}{Z_1^2} \end{bmatrix}. \tag{2.16}$$

Because the gradient of the image is not smooth, we usually calculate the derivative for each pixel when estimating the pixel-wise Jacobian with respect to photometric error [ESC14]. Correspondingly, because the camera pose belongs to the Lie group, Jacobian of all pixels on the same image are smooth so that we can calculate the derivative of the camera pose relative to the photometric error only once when solve the camera pose. This trick is known as First Estimation Jacobian trick [HMR09], and is used to ensure that the null space of camera pose Jacobian matrix will not disappear during the estimation process.

Note that equation (2.10) to (2.16) are designated for solving the pose estimation from one point. For N point problems, the Jacobian matrix will be accumulated through each point and we will finally solve for the relative pose update.

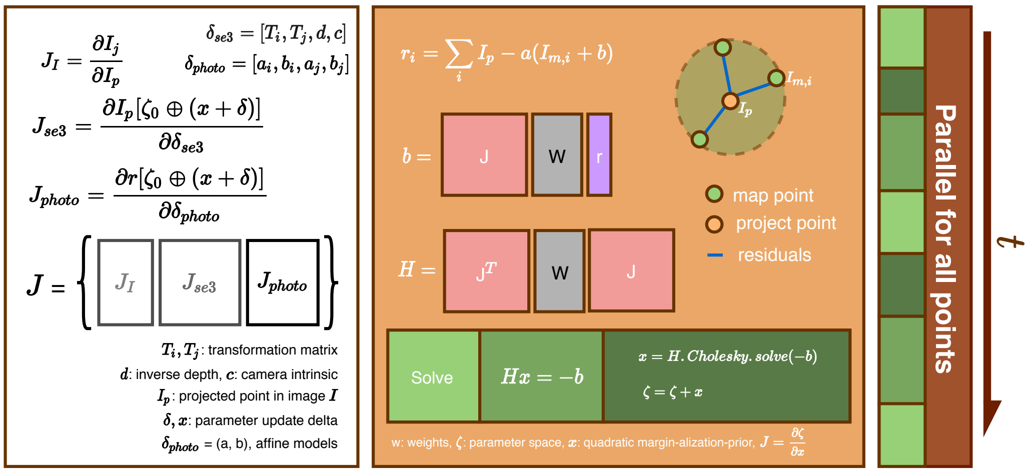

$J_I = \frac{\partial I_j}{\partial I_p}$
$\delta_{se3} = [T_i, T_j, d, c]$
$\delta_{photo} = [a_i, b_i, a_j, b_j]$
$J_{se3} = \frac{\partial I_p[\zeta_0 \oplus (x+\delta)]}{\partial \delta_{se3}}$
$J_{photo} = \frac{\partial r[\zeta_0 \oplus (x+\delta)]}{\partial \delta_{photo}}$
$J = \{ J_I \; J_{se3} \; J_{photo} \}$
T_i, T_j: transformation matrix
d: inverse depth, c: camera intrinsic
I_p: projected point in image I
δ, x: parameter update delta
δ_{photo} = (a, b), affine models
$r_i = \sum_i I_p - a(I_{m,i} + b)$
$I_{m,i}$
I_p
$b =$ J W r
$H =$ J^T W J
● map point
● project point
— residuals
Solve
$Hx = -b$
$x = H.Cholesky.solve(-b)$
$\zeta = \zeta + x$
w: weights, ζ: parameter space, x: quadratic margin-alization-prior, $J = \frac{\partial \zeta}{\partial x}$
Parallel for all points
t

Figure 2.2: Direct visual odometry method intuitive illustration.

In practice, the Jacobian of the tracking points contains three major parts, the image gradient J_I, the geometrical Jacobian J_{se3}, and the exposure affine model J_{photo} to compensate the effect of auto-exposure. As shown in Fig. 2.2, all the Jacobians are aggregated into a single Hessian matrix, as the Jacobian for each point is linear in terms of the relative pose [EKC17]. This particular property enables us to leverage parallel methods to expedite the tracking pipeline.

Combining equation (2.15) and (2.16) we can thus aggregate the Jacobian of $\delta\xi$ for each tracking point and solve the Gauss-Newton optimization as:

$$
\begin{aligned}
H &= \sum_{i \in \mathcal{P}} J_i^T W_i J_i && (2.17)\\
b &= \sum_{i \in \mathcal{P}} J_i r_i && (2.18)\\
\delta\xi &= H^{-1} b, && (2.19)
\end{aligned}
$$

where H is the point-wise Hessian matrix, b is the corresponding bias vector and W is the diagonal weight matrix, which is proportional to the inverse depth for each point. After getting the Hessian matrix and bias vector above, we can bring them into the Gauss-Newton equation to find the delta update $\delta\xi$ of the relative pose. After enough iterations, the amount of pose update will gradually tend to zero, which shows that we have found a potential optimal pose estimate.

2.3 Discussions of the direct visual odometry method

After understanding the derivation and optimization process of the direct method, we can start to discuss the shortcomings and improvements.

From the derivation in Fig. 2.2, we can see that the direct visual odometry method is very dependent on the gradient. The Gauss-Newton optimization method we employed needs to ensure that the photometric error is declining continuously during the optimization process. In practical applications, we can not ensure the photometric error is always declining. If we optimize along the image gradient, it will easily fall into a local minimum and cannot continue to optimize. Therefore, the direct visual odometry method can only be established when the camera movement is small and the initial pose estimation is close the global optimal. These conditions are actually very harsh, so we need to find an extra constraint to narrow our hypothesis space. The depth information of the image predicted by the deep neural network can provide us with such an additional constraint.

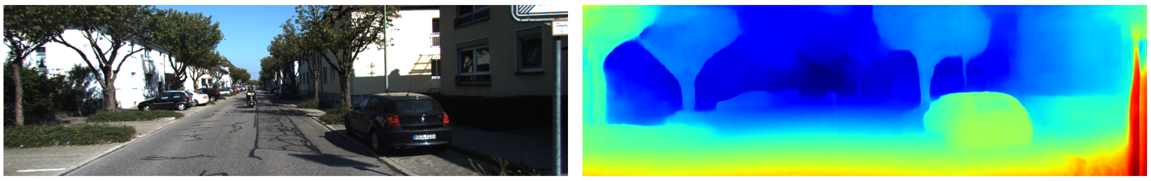


Figure 2.3: Reference frame and its depth map

Fig. 2.3 shows the reference image taken from the KITTI dataset, and its corresponding depth map predicted by MonoDepth2 [GMAFB19]. From the above figure, we can find that the depth estimation of the single frame image by the deep neural network is basically consistent with the ground truth. When the depth information of the picture is known, solving the camera pose can be simply regarded as adjusting the camera pose so that the two pictures can overlap as much as possible. This is actually the image alignment on the 2D plane. Even if the problem is simplified, a reasonable initialization condition still needs to be ensured during the pose estimation. Only when most of the pixel gradients are consistent with the correct optimization direction can we get the correct estimation results. In fact, there are still many errors in the subtleties of the depth estimation, such as tree crowns and distant objects. Therefore, we need to find a suitable method to use this depth information instead of simply substituting it into the projection formula. We will discuss related methods in detail in Chapter 4.

Thanks to the recent advance of the camera, popularization of high frame rate camera has enabled very small motion between consecutive frames. Given this fact, camera pose can thus initialized from the identity matrix.

Fig. 2.4 demonstrates the tracking process using the direct method introduced above. The direct method based pose estimation is performed from frame 00 to frame 06. With an average 10 iterations of Gauss-Newton optimization, the relative pose estimation can converge very fast since the relative pose is very small, and initialization from the identity matrix has a higher chance to converge into the global optimal given a reasonable depth estimation. The green dot in the figure is the tracking point in the reference frame. The green line shows how the point moves between two frames, in this experiment we use the depth image in Fig. 2.3. We can see from the above picture that the point close to the tree crown gradually deviates from the overall direction of movement. Due to the large error in

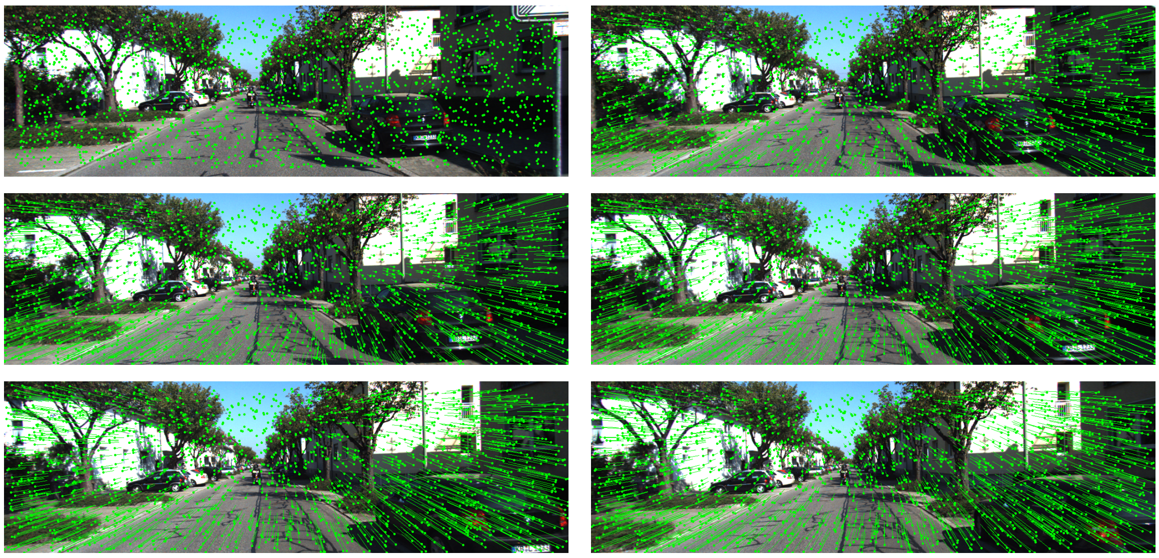


Figure 2.4: The images are a series of video frames from KITTI odometry seq. 0 (order: left top to right bottom). Green points are map points tracked by direct methods.

the depth estimation of these points, the reprojection position of these points is far away from the optimal solution. This causes the local gradient of these points to be inconsistent with the gradient of the overall direction of motion, thereby prematurely terminating the optimization of those points. This phenomenon is reflected in the short trajectory of the corresponding green dot in the figure.

Usually, motion tracking in direct visual odometry method is performed in several scale-spaces, from coarse to fine. The speedy yet coarse motion estimation will offer better motion prior for the fine scale-space estimations. This trade-off between speed and precision helps most optimization-based direct visual odometry methods maintain real-time performance as well as high tracking accuracy.

We have learned how the direct method triangulates the camera's motion from consecutive pictures in the video. In the next chapter, we will sort out some of the classic works in visual SLAM and show how the direct method can help visual SLAM evolve in realtimeness and accuracy.

3 Related Works

In this chapter we discuss related work on direct visual odometry from historical methods to current deep learning based methods.

Visual navigation technology was inspired by birds chasing their prey [Bak84]. Visual odometry term was first proposed by Moravec *et al.* [Ett01]. In the following decades, visual odometry continued to evolve and divide into two categories [AMSI16]: geometric and non-geometric methods. In the geometric method, it is divided into three sub-categories, the feature method (indirect method), the direct method, and the fusion method of the above two methods. In this thesis, we focus on comparing geometric visual odometry method which leverage the geometric property of image contents to estimate the camera poses.

There are many geometric methods for visual odometry. According to different sensors, these are divided into pure visual odometry and visual-inertial odometry that take IMU (Inertial Measurement Unit) inputs. Among them, the visual-inertial odometry involves multi-sensor fusion, so its positioning effect is much better than pure vision, and the representatives are VIO [FCDS16][UESC16], ROVIO [BOHS15] and OKVIS [LLB$^+$15]. The pure visual odometry method can be divided according to the number of sensors. There are monocular and binocular visual odometry methods. The monocular visual odometry methods includes DSO [EKC17], SVO [FPS14], and MonoSLAM [DRMS07]. The binocular visual odometry methods include Stereo-DSO [WSC17] and S-PTAM [PFC$^+$17]. Since binocular visual odometry methods have extra sensory input, they can recover the depth information from the image directly. Thus scale drift has never been a problem for binocular visual odometry. Visual odometry methods can also be divided according to the different algorithms used. Methods which use feature points and corresponding matching algorithms are called indirect visual odometry. Methods for positioning by directly comparing the image intensity are direct visual odometry. Recent advance of deep learning methods are also helping the visual odometry in terms of accuracy and robustness [WMBZL18a] [ASdG$^+$19] [KM15] [WPF19]. In the upcoming sections, we will introduce some important visual odometry techniques from the past to the present. We will also focus on the difficult problems facing direct methods and discuss current popular solutions, including introducing more mature imaging models or using deep learning to help us improve the stability and accuracy of the algorithm.

3.1 Visual odometry brief history

The research of visual odometry introduced by Moravec *et al* [Mor80] demonstrate feature correspondence to estimate 3D camera motion in 1980. Many contemporary methods [MS87][AHB37][WCR92][BC86][OMSM01][Ett01] improved this method using probabilistic approaches. When we entered the 21st century, many stable and high-performance visual features were put forward, taking visual navigation to a new height. The evolution of feature extraction algorithms is reflected in the corner detectors from Moravec and Harris [KGL10] to features that are robust to scale and angle transformations, such as SIFT [TPD08], SURF [HLFP13], ORB [GOGJ16], BRISK [JXL13] and so on. Feature extraction algorithms differ in accuracy and speed. In practice, the required features need to be selected according to needs. Chien *et al.* [CCCK16] compared and analyzed visual navigation algorithms with different features in detail. Multi-sensors also help researchers make better use of visual features in many ways. Escalera *et al.* [dlEIM$^+$16] proposed to use binocular images to guide the feature extraction area. Kottach *et al.* [KYP$^+$17] used an inertial measurement unit to filter noise in feature points. Many nonlinear systems have also been introduced to enhance visual navigation based on visual features. Davison *et al.* [DM02] and [BC86] [Hal83] use a Kalman Filter to optimize motion estimation. Webb *et al.* [WPKL07] further exploited epipolar constraint of associated features points to estimate motion states in the EKF framework. In general, the above feature point methods are mostly sparse because they are based on feature extraction, which leads them to be sensitive to local outliers and cannot perform tasks under areas without features, whereas direct method can still work as long as image gradient exists. The direct method makes up for the above shortcomings to a certain extent and gradually catches up and surpasses the feature point method in speed and accuracy.

The direct method estimates motion by directly comparing the appreance of the image. Avoiding feature extraction enables it to still work in environments where certain features are missing [EKC18]. The direct method performs image comparison globally to avoid the failure of feature association, such as in the context of repeated textures [Lab06]. The direct method can be traced back to the optical flow method proposed by Horn and Schunck [HS81] in the early 1980s. But the optical flow method is very sensitive to motion discontinuities and illumination change. Therefore Gilad Adiv [Adi85] proposed to use locally connected optical flow vectors to solve the problem of incoherent motion. In earlier ages, Clocksin [Clo78] Ullman [Ull79] and Prazny [Pra80] attempt to estimate ego-motion from optical flow. Whereas Lucas and Kanade proposed the use of image registration to estimate ego-motion [LK+81] to avoid the impact of illumination change on motion estimation. Zhou *et al.* [ZWT03] used histogram of gradient to match image appreance. The above method is not good at estimating the rotation between images. Comport specifically proposed the use of binocular vision for 6 DOF motion estimation [CMR07]. He proposed to use the method of minimizing image intensity to estimate the relative motion pose. Lovegrove *et al.* [LDIG11], also proposed the use of image registration to estimate pose and the use of special textures such as planes in the image [15]. After entering the 21st century, with the advancement of camera imaging technology, researchers proposed to use the gradient of image intensity to estimate the relative motion of the camera [GBG12][TC11]. In recent years, Engel *et al.* [ESC14] proposed the sparse direct method and sparse bundle adjustment to optimize the photometric error. It avoids the geometric prior in the feature point method and uses the sliding window method to improve stability. This thesis inherits the work of Engel *et al.* [EKC17] and continues to use the direct method for relative pose estimation in monocular vision. The difference is that we use deep learning to solve the ill-posed problem in monocular vision, that is, scale uncertainty and drift.

3.2 Monocular visual odometry method

Monocular visual odometry method based on the feature point method has long been (until now) regarded as the mainstream method. It runs stably, is insensitive to lighting and dynamic objects, and is currently a relatively mature solution. Among the representative works include ORB-SLAM [MAMT15] and PTAM [KM07], as shown in Fig. 3.1. These algorithms first select representative points from the image. These points will remain unchanged after a small change in camera angle, so we can find the same point in each image.

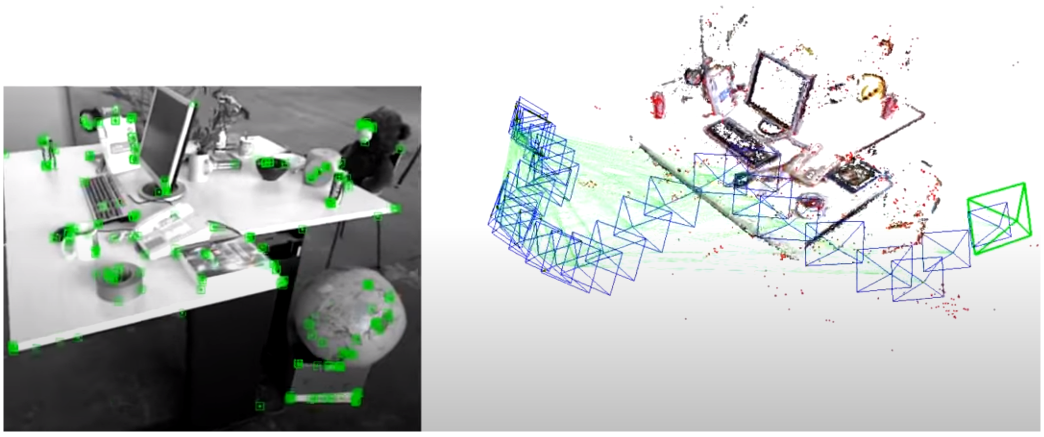


Figure 3.1: Feature based visual odometry method (ORB-SLAM [MAMT15]).

The visual odometry method can estimate the camera's pose and the position of each points by finding correspondence between points. However, the common disadvantage of the feature point method is that the environment must contain texture with reliable features. When the scene does not contain enough feature points in the absence of texture, it will affect the stability of the algorithm and even invalidate the algorithm directly. In addition, sparse feature points can also cause tracking loss. The most important thing is that algorithms for calculating feature points are generally very time-consuming, making such algorithms difficult to transplant to embedded devices or robots. Therefore, people need to consider skipping complex feature point calculations while trying to avoid dependence on textures. At this time, the direct method has begun to receive widespread attention.

Monocular direct visual odometry is primarily deployed on RGB-D camera [KHK13], but the sensor is built for indoor scenarios. In order to get rid of sensor limitations, the depth filter [FPS14][ESC13] was proposed to model the depth with probabilistic. Jakob *et*

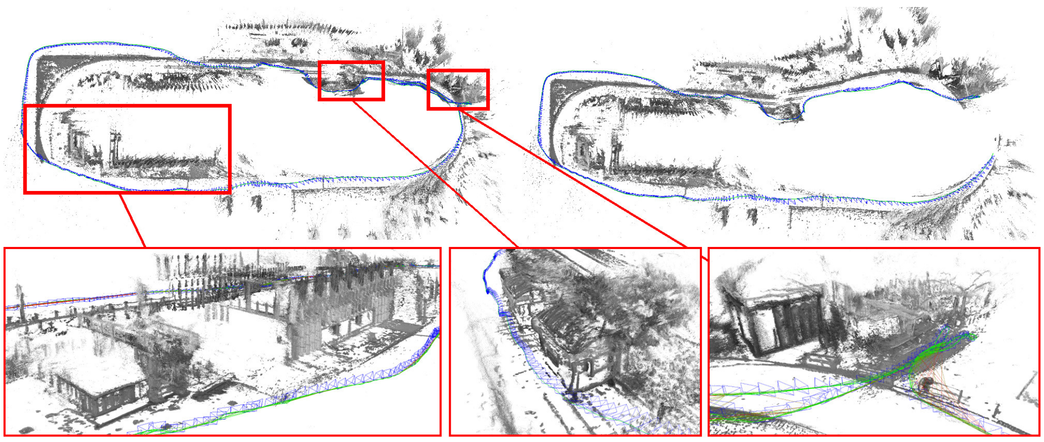


Figure 3.2: Direct visual odometry method (LSD-SLAM [ESC14]).

al. [ESC14] further parameterized depth into the optimization back-end. Depth and camera pose in the monocular visual system are coupled, which leaves one degree of freedom in the null-space and results in an ambiguity of scale. This deficiency is often remedied by integrating more sensors into the system - for instance, an IMU, which provides constraints in the form of inertial measurements. The stereo version of DSO [WSC17] and inertial measurement [VSUC18] have substantially improved the scale of estimation. However, the cost of adding a new sensor and the multiple signal alignment makes their deployment infeasible for mobile robots with limited hardware.

3.3 Deep learning for visual odometry method

Attempts to solve visual odometry with the help of deep learning methods have received more and more attention. SfMLearner [ZBSL17] started the unsupervised architecture based on back-warping amongst consecutive image frames. DVSO [YWSC18] then developed the same idea on left-right warping and generated virtual stereo pairs to estimate monocular depth and camera poses. Yet the lack of normalization in their depth predictions increased the likelihood of divergence. DDVO [WMBZL18b] addressed this issue in their model by normalizing the output of the depth CNN before feeding it into the loss function.

Furthermore, they [ZBSL17][YWSC18][WMBZL18b] tried to integrate the direct image alignment backend and the depth estimation pipeline to jointly optimize their pose and depth networks, which couple with a shared encoder. Monodepth2 [GMAFB19] further investigated the occlusion effect and adopted a coarse-to-fine trick by predicting depth and pose in different scale-spaces. All the above methods propose an independent pose network to produce the consecutive frame warping transformation matrix. The generalizability of the pose network remains an issue regardless of the training procedure, which is not the case with traditional visual odometry methods. Consequently, DDSO [ZTS19] and CNN-SVO [LAM$^+$19] utilize the depth prediction results from a deep CNN to help initialize the visual odometry system, and then use traditional geometric methods to estimate camera poses. Note that while the initialization plays an essential part in VO systems, scale-drifts will still occur during the tracking thread. Both DDSO and CNN-SVO assume that the predicted depths are accurate across the whole image, but in actuality, depths predicted at further distances tend to be noisy. This uncertainty can be problematic, as the relatively long life span of distant map points more heavily contribute to pose estimation in a local sliding window. In this work, we carefully model uncertainty and propose an uncertainty-aware inverse depth variance propagation and fusion.

4 Depth Estimation for Direct VO

In this section, we fully integrated a deep CNN based depth estimator into the pipeline of direct visual odometry. At the same time, we utilize the sparse depth results extracted from successfully tracked map points to refine the depth predictions further. We also briefly discuss the relationship between the residual pattern and local convexity of photometric loss to further improve the robustness of our direct visual odometry system.

4.1 Dynamic Point Selection

Direct visual odometry samples the high gradient points in the host frame, and then guesses the initial relative camera pose between the host frame and the target frame. Usually, an identity matrix was chosen to estimate the initial depth of the tracking points. Given both initial pose and depth, one can project each tracking point from the host frame into the target frame. The direct pixel intensity error will guide the optimizer to adjust both relative pose and point-wise depths until it converges to a local minima. Given the point-wise depth predicted from a deep CNN, we can directly solve the camera pose efficiently. We begin by describing our strategy for selecting the initial tracking points in the system. Then we fully develop the system components in three major pipelines: initializing, tracking, and backend optimizing. In the backend part, we briefly discussed how depth estimation is used to help marginalization and loop closing. We also extended our methods to refine the depth prediction further using VO sparse map points.

Point selection is a fundamental component of any DVO system and holds a strong bearing on initialization and tracking robustness. Gradient-based direct methods favor the selection of points in high gradient regions, such as corners and edges, as they contribute more to the photometric reprojection error, and thus, can boost the efficiency of bundle adjustment. However, the potential clustering of points near these corner or edge-like features may expose the system to high estimation error when most high gradient features are grouped in the same sub-section of a given image (Fig. (7) of [EKC17]). To reduce estimation error under such circumstances, we propose a new point selection method, which balances the needs for a more even spread of points with sampling near high gradient features (as shown in Fig. 4.1). The leftmost image in Fig. 4.1 is the original RGB input. The middle image demonstrates our point selection result (colored by distance). The rightmost image shows the DSO point selection result. Unlike DSO, which selects points depending on the local image gradient, we increase the sample rate in low gradient regions to maintain an even spread of tracking points in the image space. Note that we select points from the whole image. Point distributions in Fig. 4.1 are all from cropped images. We show the original image for literature consistency. We cropped 15% of the top part in all input images since the depth estimation is very noisy on the sky.

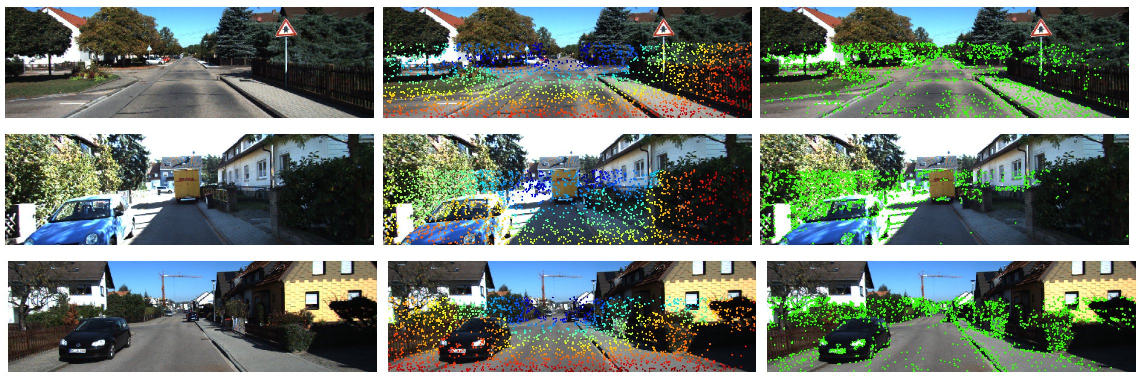

(a) RGB Image (b) Our point selection strategy (c) DSO point selection strategy

Figure 4.1: Initial tracking point selection.

As mentioned above, a good tracking method requires a fairly strict initialization for second-order optimization. We pick the initial tracking point following a dynamic gradient-based sample technique. As shown in Fig. 4.2, our dynamic point sampling method selects tracking points according to both the local gradient and global distribution in different scale-spaces. Here is the full pipeline of the gradient-based point selector:

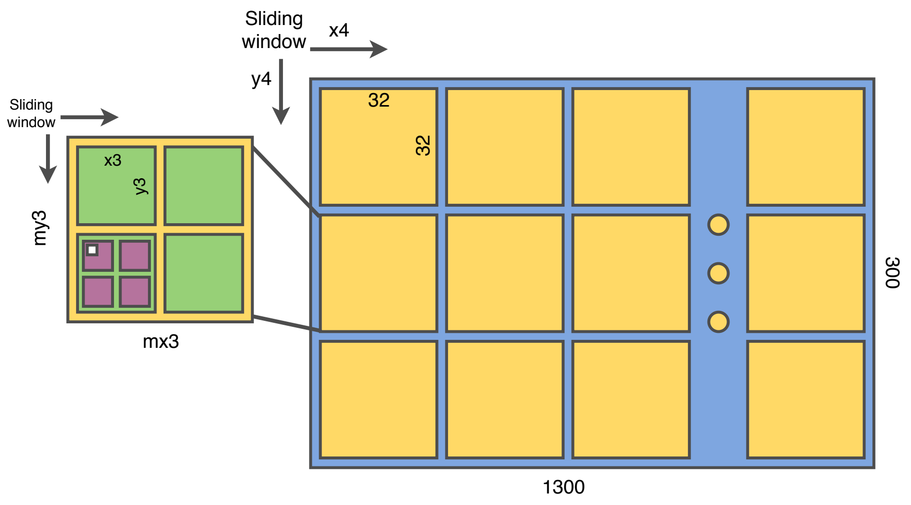


Figure 4.2: Our dynamic pixel selector.

- Calculate horizontal image gradient I_x, and vertical image gradient I_y, and get $I_x^2 + I_y^2$ as "absSquaredGradient" map (same size as original image). Then cut the image into 32x32 small patches and accumulate their gradients into histograms (HOG, 50 bins).
- Take the first quantile of gradients (histogram of the gradient is size of 1024, take 90, which is top 10 $\%$ highest gradients) as the HOG threshold of the scaled image (x32 smaller) and smooth the threshold map using its mean value.
- Finally, the threshold (x32 smaller) map is used to guide whether a pixel point should be selected or not according to its gradient.

After obtaining a group of points that can effectively cover the entire image, we can officially start tracking and maintaining these points. Some of them will gradually converge and become map points during the optimization process, while others will be kicked out as outliers. Next, we will introduce this process with a complete pipeline, and expand the details in the following sections.

4.2 Overview of our system design

We introduce the complete pipeline of our system in this subsection. As shown in Fig. 4.3, the system is divided into two major parts: a) tracking; b) mapping. The deep CNN depth estimator is used in the initialization and reprojection processes.

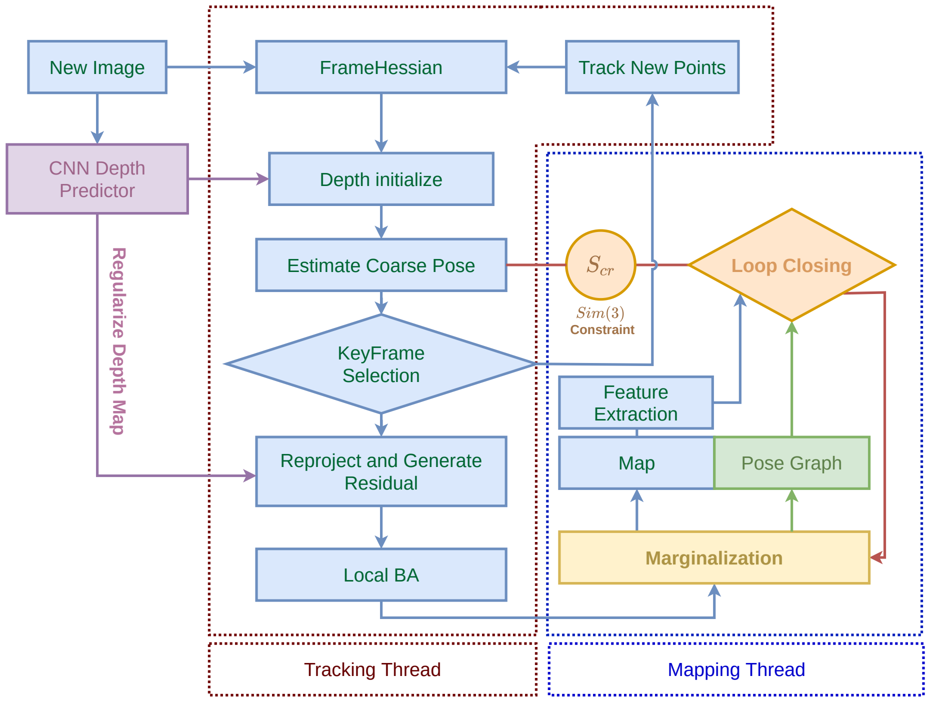


Figure 4.3: CNN-DVO pipeline

Inspired by DSO [EKC17] and LDSO [GWDC18], we adopt the same keyframe based sliding window approach. Within the local siding window, the $SE(3)$ keyframe pose and inverse depth are parameterized and composed into a minimization problem as formulated in [GWDC18], Eq. (1). One key idea proposed in this thesis is to utilize the deep CNN estimated depth in the current camera image to estimate camera pose in the next frame. We adopt the pre-trained MonoDepth2 [GMAFB19] mono+stereo, 640x192 configuration as our depth predictor which helps to refine the pose estimation as shown in Fig. 4.3. The loop closing part including pose graph build, map database build, loop detection and loop closing, the loop detection and closing are also used to guide the marginalization of keyframes and offer extra $Sim(3)$ constraint on estimating new frame poses.

By continuously propagating the depth into each tracking frame and refining it in the local bundle adjustment, we can easily lock down the inverse depth variance upper bound. We perform Gauss-Newton optimization on the Jacobian matrix containing all selected points to refine their optimal reprojection location in the new frame and recover the new inverse depths in the epipolar line [ESC13]. Given the inverse depth prior, the search interval is limited by $d \pm 2\sigma_d$ where d is the inverse depth, σ_d is the standard deviation which is initially set as linear combination of the inverse depth prior [LAM$^+$19] and observation variance [ESC13],

$$\sigma_d^2 = \lambda(\frac{d'}{6})^2 + (1-\lambda)\sigma_{d,obs}^2. \tag{4.1}$$

Here λ is the weight (empirically set as 0.54), d' is the predicted inverse depth from the deep CNN, $\sigma_{d,obs}^2$ is the observation variance of the inverse depth which is proportional to the multi-view stereo disparity error. The empirical setting provides adequate room for noisy depth predictions to converge.

Following the regularization trick in LSD-SLAM [ESC14], we incorporate the depth prior into our photometric equation as a depth residual, which penalizes the deviations in inverse depth between keyframes, and properly scales the estimated transformations between them. The residual in turn is defined as:

$$r = (I_j[p'(T_i, T_j, d', c)] - b_j) - \frac{t_j e^{a_j}}{t_i e^{a_i}}(I_i[p] - b_i), \tag{4.2}$$

where T_i, T_j are camera poses, d': the depth prior, c: camera intrinsic, a_i, a_j, b_i, b_j: affine brightness coefficients, I_i, I_j are two keyframes, p' in I_j are projected points from p in I_i given the transformation above. The windowed optimization part is similar to DSO; please refer to Section 2.3 in [EKC17] for implementation details.

We optimized the loop closure and pose optimization backend of LDSO [GWDC18]. Since we already have a depth prior in each keyframe, there are no longer any 2D geometric constraints, and the cost function is hence simplified to:

$$\mathbf{E}_{loop} = \sum_{q_i \in Q} ||\Pi(\mathbf{S_{cr}}\Pi^{-1}(p_i, d_{p_i})) - \Pi^{-1}(q_i)||_2, \tag{4.3}$$

Q are the matched features in the current keyframe, Π and Π^{-1} are the projection and backward projection functions, S_{cr} is the target $Sim(3)$ constraint to be estimated from the cost function above. However, taking each keyframe and their tracked map points in the global mapping database is a heavy burden on the system. To remedy this, we follow the keyframe selection trick in SPTAM [PFC+17]: with a current pose estimated in the sliding window, a frame is selected to be a keyframe if the tracking points consist of less than 90$\%$ of those from the last keyframe in the pose graph. The keyframe is then pushed into the sparse pose graph and offline loop closure is performed in the mapping thread.

In loop closure, we extract the BRIEF (Binary Robust Independent Elementary Features) descriptors on each keyframe and maintain a DBoW [GLT12] database (an open source C++ library for indexing and converting images into a bag-of-word representation) for loop detection query, then initialize the transformations by applying RANSAC PnP (Random sample consensus, Perspective-n-Points), and optimize the $Sim(3)$ transformation through 3D-2D geometric constraints. We only update the optimized pose in the front-end thread (visualizer) to avoid corrupting the local sliding window by modifying the backend pose graph, and to keep the tracking thread running efficiently.

A notable feature of our method is the ability to recover the scale of absolute pose constraints between the loop candidate and the current frame. During global pose optimization, we fix the pose estimates of all reference frames and apply the aggregated correction transformation updates only in the front-end visualizer. Since the pose Hessian prior is carried on in each frame in the local sliding window, modifying the pose in the backend will corrupt the local windowed bundle adjustment, and thus, for thread safety and tracking robustness, the global map optimization only occurs upon manual triggering or when tracking concludes.

To achieve dense mapping, we further refine the dense depth map by propagating the sparse depth extracted from the successfully tracked 3D map points. With the help of spatial propagation network introduced in [LDMG+17], we managed to precisely map the environment and achieve state-of-the-art result on public visual odometry benchmark.

4.3 Depth estimation for initialization

Monocular depth estimation: Monocular depth estimation has been well explored by the vision community, and many existing methods show impressive results on public datasets such as KITTI and CityScapes. The mainstream architectures are built from the encoder-decoder structure [GMAB17] (Fig. 4.4). As shown in Fig. 4.4, the RGB image I_t is fed into a U-Net [RFB15] shape network and predict the depth per pixel d_t. Similar to the semantic image segmentation problem, the per-pixel loss is accumulated through photometric error raised by pixel-wise warping. This transformation will inevitably introduce the pose network (Fig. 4.5). As shown in Fig. 4.5, two consecutive RGB images $I_t, I_{t'}$ are fed into the pose network, the network is consist of a convolution neural network and fully connected layers and it predict the 6 DoF camera pose $\xi_{t\to t'}$. Similar to [WMBZL18a] which adopt differential direct visual odometry methods to train the pose network in an unsupervised fashion, approaches like [ZGSW$^+$18] [ZBSL17] all share the same back-warping trick to obtain photometric loss. On the contrary, [RJB$^+$19] and [GMAFB19] all work with a supervised pose network. Since we are not interested in estimating camera pose with a pure deep learning module, we employed Monodepth2 as our default depth estimator. We use pretrained Monodepth2 [GMAFB19] checkpoints, which have been fine-tuned for KITTI raw stereo data and perform the real-time depth inference on GPU ($\sim$20 ms). The predicted depth priors are disparity maps, which are analogous to inverse depth scaled by focal length and baseline. In practice, the disparity maps were scaled by 0.1 to ensure numerical stability.

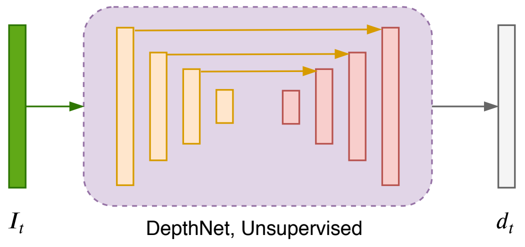


Figure 4.4: Unsupervised encoder-decoder based depth prediction.

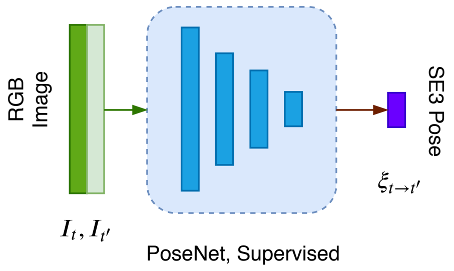
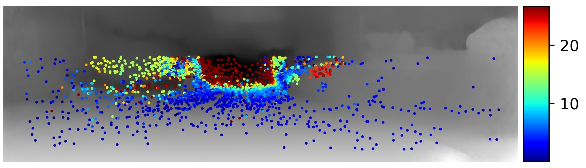


Figure 4.5: Supervised CNN-pose network.

We observed relatively high estimation errors from the deep CNN at mid-to-long distances, as depicted in Fig 4.6. The top image is the original RGB input, the middle image is the dense depth predicted from Monodepth2 and is bilinearly upsampled by 2x, the bottom image is the depth error between predicted dense depth map and successfully tracked map points. The depth error is in meters. This is a result of predicting disparity maps (equivalent to inverse depth map), where long-range pixel-wise depths are encoded as small decimal valued inverse-depths (disparities), which contribute very little to the final loss. This source of high estimation error is prevalent in many methods that predict disparity maps [GMAFB19][ZBSL17][WMBZL18b], and is not the case in supervised learning approaches that use ground truth depth labels.

(a) RGB Image



(b) Inverse Depth and Tracked Map Point Depth Error

(c) Inverse Depth from Deep CNN

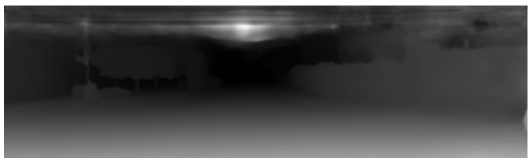

(d) Refined inverse depth from sparse tracking points

Figure 4.6: Depth estimation is refined from tracked map points.

Furthermore, the low disparity of the sky usually leads to noisy depth estimates from the deep CNN in those regions. This noise may corrupt the depth prior, and hence, we remove all tracked points in the upper $\sim 30\%$ of the image, as illustrated in Fig. 4.1. We use the finely predicted parts of images to initialize the depth filter and to track map points - this process is thoroughly explained in the follow sections.

Depth estimation for initialization: We employ a depth filter to initialize our inverse depth estimation for each selected immature point. We express the optimal inverse depth as a function of our depth prior and geometric projection inputs:

$$d^* = d(I_0, I_1, d', \xi, \pi). \tag{4.4}$$

Here d^* is the optimal inverse depth, d' is the inverse depth prior defined in Eq. (4.1), ξ is the relative transformation matrix and π is the projection function. The error variance σ^2_{d*} is thus given by:

$$\sigma^2_{d*} = \alpha^2 \left(c_d + c_d \frac{J_d \Sigma J_d + J_d' \Sigma J_d'}{J_d \Sigma J_d} + \sigma^2_d \right). \tag{4.5}$$

Where c_d is the normalization constant (empirically set to 0.2), J_d is the Jacobian of d ($J_d = [dx, dy]^T$, as shown in Fig. 4.7). In the same figure, f is the focal length, I_* are the images, d' is the inverse depth prior, σ_d^2 is the inverse depth variance, σ_{d*}^2 is the optimal inverse depth variance. d_x and d_y are search region dimensions on the epipolar line. The depth prior narrows the search region and makes our method tenable to large baseline stereo problems, depicted in Fig. (3) of [ESC13] (i.e. strong rotation motions exemplified in Fig. (7) of [WSC17]), J_d' is the conjugate Jacobian of d: $J_d' = [dx, -dy]^T$, Σ is the 2×2 input-error covariance ($\Sigma = [I_x, I_y]^T[I_x.I_y]$), α is defined in Eq. (11) in [ESC13], and σ_d is defined in Eq. (4.1). We initialize our given inverse depth prior to $\mathcal{N}(d', \sigma_d^2)$, and perform a Gauss-Newton optimization to shrink the inverse depth upper bound for each candidate map point:

$$r_{id} = \sum_{i\in\mathcal{P}_i} ||I_1[\pi(p_i, d, \xi)] - e^a I_0(p_i) + b||_\gamma. \tag{4.6}$$

Here r_{id} is the inverse depth residual, γ is huber norm, $\mathcal{P}_i$ is the residual pattern, π, d and ξ are defined in Eq. (4.4), a, b are affine coefficients defined in Eq. (4.2). Likewise, the Jacobian of inverse depth can be derived as implemented in [EKC17]:

$$J_{id} = I_x d_x + I_y d_y. \tag{4.7}$$

Where I_x, I_y are image gradients at a given pixel location. d_x and d_y are the search region dimensions along the epipolar line, as shown in Fig. 4.7.

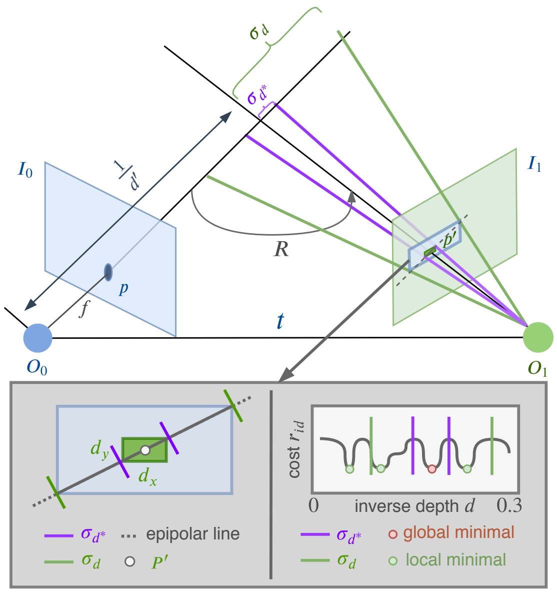


Figure 4.7: The uncertainty propagation of inverse depth.

Since the parameterized inverse depth is a scalar, the Hessian of the inverse depth and error term are also scalars: $H_{id} = J_{id}^2$, $b_{id} = -J_{id}r_{id}$, respectively. The optimal pixel coordinate p will be updated through $p* = p - \frac{b}{H}$, and the inverse depth is consequently recovered from p according to the projection equation in [ESC14]. Note that the optimizer will converge into the optimal pixel coordinate with respect to the inverse depth, thus, the inverse depth can be recovered from the projection equation as illustrated in [CDM08].

Not only does the depth prior shrink the range of the search along the epipolar line, but it also initializes the starting point in the neighborhood of the local minimum, which increases the convergence rate of the second-order optimizer. Further, the observation variance fused from the depth prior can then be propagated to new frames to infer the new inverse depth and camera pose.

4.4 Depth estimation for tracking

Ideally, the predicted depth from the deep CNN should be precise everywhere. However, we observe that the error near high gradient areas such as object boundaries is often quite large, despite the introduction of an edge-aware term in the commonly used smoothness loss. This extra noise is modelled in the inverse depth variance of new tracking frames, once the camera position of the frame has been estimated. The new inverse depth d_1' is defined as:

$$d_1'(d_1, d') = \mathcal{N}\left(\frac{\sigma_{d_1}^2 d' + \sigma_{d'}^2 d_1}{\sigma_{d_1}^2 + \sigma_{d'}^2}, \frac{\sigma_{d_1}^2 \sigma_{d'}^2}{\sigma_{d_1}^2 + \sigma_{d'}^2}\right), \tag{4.8}$$

where d_1 and σ_{d_1} are defined in Eq. (14) and Eq. (15) from [ESC13]. Note that this is equivalent to the update step in a Kalman filter where the noisy observation $\mathcal{N}(d', \sigma_{d'}^2)$ is fused with the geometrical propagation prior $\mathcal{N}(d_1, \sigma_{d_1}^2)$.

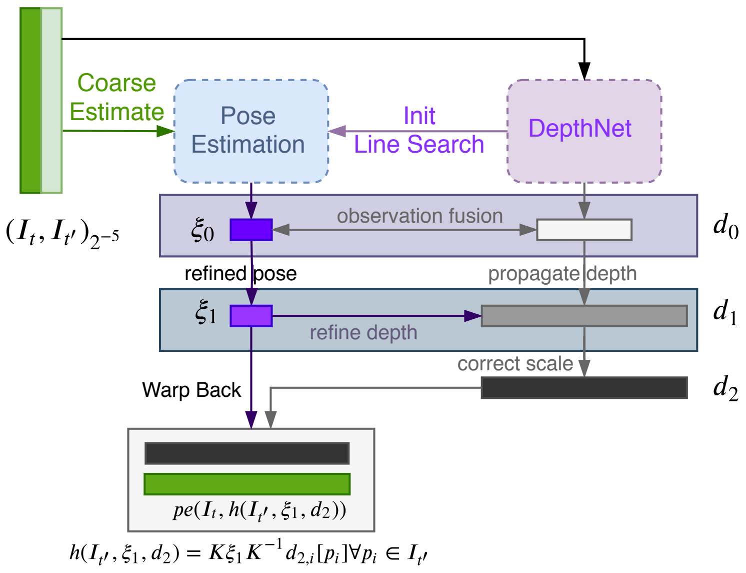


Figure 4.8: Depth estimation for tracking pipeline.

Our tracking pipeline is illustrated in Fig. 4.8. Low resolution depth is predicted as a coarse prior to initialize the coarse camera pose, the inverse depth filter is then applied to optimize the sparse depth map. I_* are images, ξ_* are camera frame poses, d_* are sparse depth maps, $pe(\cdot)$ and $h(\cdot)$ are photometric error and projection function defined in [WMBZL18b]. Please refer to Fig. 4.5 for pose inference, and Fig. 4.4 for depth estimation deep neural network architecture.

We can regard the tracking problem in the direct method as a local bundle adjustment. Bundle adjustment, in turn, is the least square optimization problem which finds the best relative pose that minimizes the reprojection error. To this end, we can throw all the geometric variables into one big vector x defined as following:

$$x = \Big[\xi_1, \xi_2, \xi_3, ..., \xi_m, p_1, ..., p_n\Big]^T. \tag{4.9}$$

Accordingly, the delta update of geometric variable x is composition of two parts: δx_ξ and δx_p, the objective function in turn is:

$$\frac{1}{2}||e(x+\delta x)||^2 \approx \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n}||e_{ij}(x) + J_{\xi,ij}\delta\xi_i + J_{p,ij}\delta p_j||^2. \tag{4.10}$$

We have derived J_ξ in equation (2.16) and J_p as parameterized inverse depth in equation (4.7). Rewrite the camera pose vectors as x_c:

$$x_c = \Big[\xi_1, \xi_2, \xi_3, ..., \xi_m\Big]^T \in \mathbf{R}^{6\times m}. \tag{4.11}$$

Note that equation (4.10) is the sum of many small quadratic equations generated from each tracking point, and all those equations are sharing the same pose prior. That is why all point-wise Jacobians can accumulate in their frame Hessian of camera pose. Here we write the Jacobians as long vectors and stack together to illustrate the same idea in a more orderly way. From equation (4.10) we can also see that camera pose Hessian matrix can be approximated as quadratic of Jacobians, thus can be written as the following format:

$$H = \begin{bmatrix} J_\xi J_\xi & J_\xi J_p \\ J_\xi J_p & J_p J_p \end{bmatrix}. \tag{4.12}$$

However, it will be diffult to solve for the inverse of the big H matrix. Luckily, the Schur complement (an non-overlapping domain decomposition method, see also in [Zha06]) is very effective in solving the big sparse Hessian matrix. Prior to solving the Schur complement, we should take a look at the structure of the Jacobian matrix, as shown in Fig. 4.9.

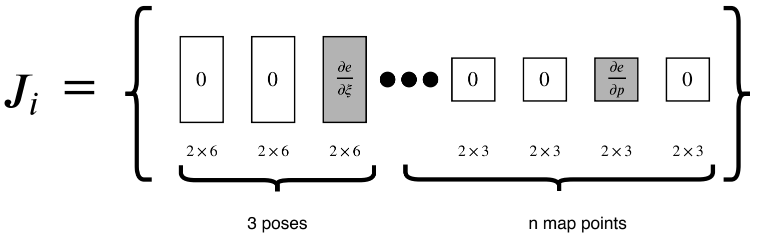

J_i =
0
0
$\frac{\partial e}{\partial \xi}$
0
0
$\frac{\partial e}{\partial p}$
0
2×6
2×6
2×6
2×3
2×3
2×3
2×3
3 poses
n map points

Figure 4.9: Jacobian matrix in local bundle adjustment.

Here $0_{2\times 6}$ means a 2 by 6 zero matrix, since $\frac{\partial e}{\partial \xi}$ is essentially a 2×6 matrix. 6 means R, t which is 6 DoF (Degrees of Freedom), the error is $e = z' - h(\xi, p)$, where e is the error of 2D point position reprojection error. From Fig. 4.9, we can see that this Jacobian matrix is mostly zero, except two non-zero partial derivatives. Those zero blocks mean that the error e does not correlate with those poses and landmark points. Thus when applying $H = J^T J$, we can see that sparsity from J will directly contribute to H.

If we represent camera poses and landmark points as nodes, the edges that connect each other should be the non-zero entries on Jacobian matrix (as shown in Fig. 4.10).

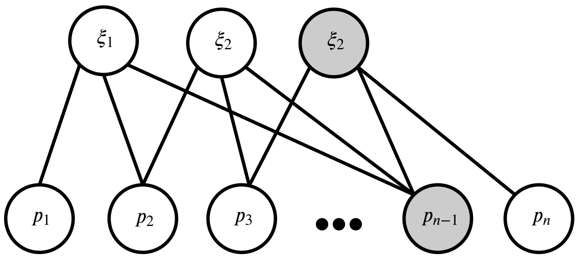


Figure 4.10: Co-visibility graph in local bundle adjustment.

When all the Jacobian matrix for each point gets stacked into the Jacobian vector $J_{\xi,p,ij} \in \mathbf{R}^{2\times 6}$. The Jacobian vector will be a big matrix as shown in Fig. 4.11.

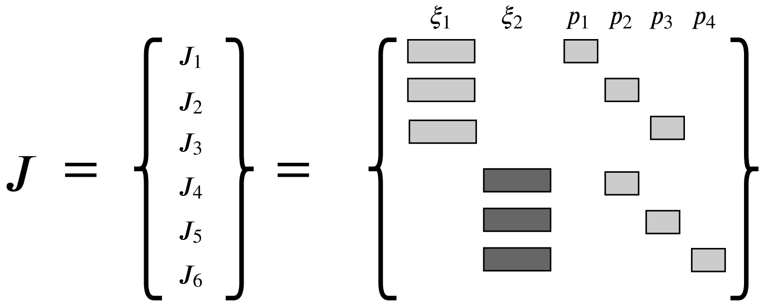


Figure 4.11: Stacked Jacobian vector.

The gray areas are the non-zero parts, which means there is a partial derivative with respect to that parameter. The longer gray rectangle is camera pose, the gray square is landmark point.

The Hessian matrix, as you can see, is the same shape as **adjacency matrix** (except the diagonal blocks). We can regard those off-diagonal non-zero blocks as constraints between poses and points. Thus we can leverage the sparsity of the H matrix and solve pose, point delta updates with Schur complement. Consider the following Gauss-Newton equation:

$$\begin{bmatrix} F^TF & F^TE \\ E^TF & E^TE \end{bmatrix} \Delta x = [F, E]^T e. \tag{4.13}$$

We rewrite H matrix with 3 blocks B, C, E:

$$\begin{bmatrix} B & E \\ E^T & C \end{bmatrix} \begin{bmatrix} \Delta\xi \\ \Delta p \end{bmatrix} = \begin{bmatrix} v \\ w \end{bmatrix}, \tag{4.14}$$

B is the diagonal block matrix, which represent camera pose, C is also diagonal block matrice, each block is 3x3. Since diagonal block matrice inverse complexity is way easier than the normal matrix, we can only do inverse on those diagonal blocks. Now, let's do Gauss elimination for the equation above:

$$\begin{bmatrix} I & -EC^{-1} \\ 0 & I \end{bmatrix} \begin{bmatrix} B & E \\ E^T & C \end{bmatrix} \begin{bmatrix} \Delta\xi \\ \Delta p \end{bmatrix} = \begin{bmatrix} I & -EC^{-1} \\ 0 & I \end{bmatrix} \begin{bmatrix} v \\ w \end{bmatrix}. \tag{4.15}$$

Reorder equation (4.15), we can get:

$$\begin{bmatrix} B - EC^{-1}E^T & 0 \\ E^T & C \end{bmatrix} \begin{bmatrix} \Delta\xi \\ \Delta p \end{bmatrix} = \begin{bmatrix} v - EC^{-1}w \\ w \end{bmatrix}. \tag{4.16}$$

It's obvious that first line of this equation is irrelevant to Δp. So we can use first line of the equation to solve camera pose:

$$[B - EC^{-1}E^T]\Delta\xi = v - EC^{-1}w, \tag{4.17}$$

and use camera pose we just solved above (equation (4.17)) to further solve Δp:

$$\Delta p = C^{-1}(w - E^T\Delta\xi). \tag{4.18}$$

Many direct visual odometry methods use the Schur complement to solve this Hessian matrix, e.g., OKVIS [LLB$^+$15], DSO [EKC17]. These type of methods require a bundle adjustment for each frame of the input image to prevent cumulative errors. However, such operations will seriously affect the real-time performance of the algorithm when the Hessian matrix is large, so they must use some techniques to maintain the sparsity of the above matrix. For example, use local bundle adjustment which only consider a small set of frames and reduce the tracking point size.

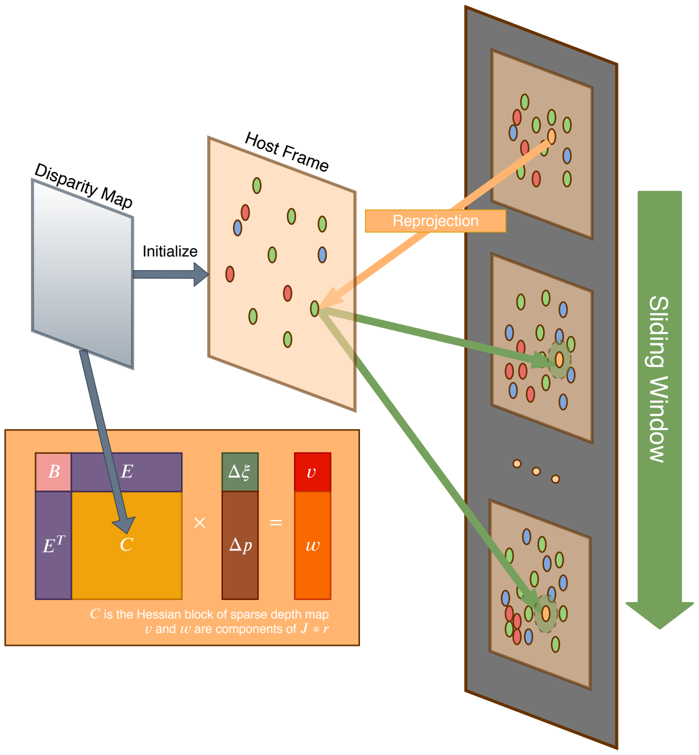


Figure 4.12: Depth prediction for the Schur complement.

Since C consists diagonal blocks, the inverse of C is very easy to solve. Note that all the Schur complement steps are performed on the matrix representation. In practice, we usually iterate over each point-wise Jacobian and accumulate the results for solving the camera pose Hessian matrix. The inverse depth prior can blend in the iteration of the Schur completion to aggregate for the Hessian matrix (expressed in equation (4.22)). As shown in Fig. 4.12, the inverse depth is used to estimate the reprojection of the points. The Hessian matrix generated by these reprojections contains the inverse depth update information of each map point. The inverse depth of these points will be updated in the optimization iterations. The point of convergence will move to the upper left corner of the C matrix, while the point of non-convergence will sink to the right bottom corner and will be marginalized in the next update (expressed in equation (4.22)).

4.5 Depth estimation for VO backend

Depth prior in Loop closing: We build our loop closure and pose optimization backend from LDSO [GWDC18] and feed in the extra depth prior measurements.

Since we already have the converged inverse depths of active map points in the library, along with a strongly estimated depth prior from the deep CNN, we can only apply 3D-3D correspondences; the $\mathbf{Sim}(3)$ transformation can be cast into $\mathbf{SE}(3)$ by scaling the translation part by the depth prior. Hence, the cost function is simplified as in equation (4.3). However, storing each keyframe and their tracked map points in the global mapping database is a heavy burden on the system in terms of time and memory. To avoid this, we select a keyframe if the tracked feature points drop below a predefined threshold portion of those from the previous keyframe in the pose graph.

Depth prior in Marginalization: We apply marginalization for three tasks: (a) to solve camera poses, (b) to eliminate outlier map points, and (c) to remove redundant keyframes. Note that the Jacobians of geometry parameters ((T_i, T_j, d, c)) with respect to two frames' poses are linearly related. The overall Hessian matrix can be decomposed into two parts: one pose Hessian which is very small, and one point Hessian vector which is a large vector containing all the inverse depth Jacobians in each entry. The pose Hessian is aggregated through each point Hessian as follows:

$$\hat{H}_\xi = \sum_{i \in \mathcal{P}_g} H_\xi - H_{\xi,p_i} H_{p_i}^{-1} H_{p_i,\xi}. \tag{4.19}$$

Where $\hat{H}_\xi$ is the updated hessian matrix of camera pose. $\{\cdot\}$ means set of all points, same representation as in equation (4.20). Bias $\hat{b_\xi}$ is also updated through the following equation:

$$\hat{b_\xi} = \sum_{i \in \mathcal{P}_g} \{b_\xi - H_{\xi,p_i} H_{p_i}^{-1} b_{p_i}\}. \tag{4.20}$$

After the linear aggregation, the pose update δ can be solved from:

$$\delta = \hat{H_\xi}^{-1} \hat{b_\xi}. \tag{4.21}$$

And then used to update the current state: $\xi' = \delta + \xi$. Note that the plus operation here is defined in Lie manifold. In practice, $H_{\xi,p_i} = J_\xi J_{p_i}$ is stored as a vector of Jacobians where each Jacobian entry corresponds to a point, and is aggregated to solve $\hat{H}_\xi$, where J_ξ is the Jacobian of the camera pose. According to Eq. (4.7), the Jacobian of points (parameterized as inverse depth) is defined as:

$$J_{p_i} = f_x(t_1 - ut_3)I_x \frac{d'_T(p_i)}{d'_R(p_i)} + f_y(t_2 - vt_3)I_y \frac{d'_T(p_i)}{d'_R(p_i)}. \tag{4.22}$$

Where $fx(t_1 - ut_3)$, $f_y(t_2 - vt_3)$ are proportional to d_x, d_y defined in Eq. (4.7), u, v are pixel coordinates in the target frame and t is the relative translation between the reference frame and target frame, and d'_R and d'_T are inverse depth priors from the reference frame and target frame. Note that these depth priors are used to initialize the Jacobian of points following the First Estimation Jacobian trick [HMR09], as we assume the depth Jacobian is smooth within the local tangent space of current parameter state.

The inverse depth in turn will be updated for each point in the host frame based on the optimized camera pose ξ':

$$\delta_{p_i} = H_{p_i}^{-1}(b_{p_i} - H_{\xi,p_i}^T \xi'). \tag{4.23}$$

For each point p_i, the inverse depth $d_{new}[p_i]$ is thus updated by $d_{new}[p_i] = d'_R[p_i] + \delta_{p_i}$. Here, $d'_R[p_i]$ is the inverse depth of point p_i in the reference frame, and $d'_T = d'_1$ is defined in Eq. (4.8).

Since the tracked points carry the consistent depth information, we further leverage this artifact information to refine our depth prediction result by applying a sparse-to-dense depth refinement module on our system. The spatial propagation module was initially introduced for semantic segmentation refinement for sharing the weights along multiple filter paths. This module is also fit for sparse depth refinement. Considering that the propagation iterations are simply updating one image into another while maintaining the original image's geometric details.

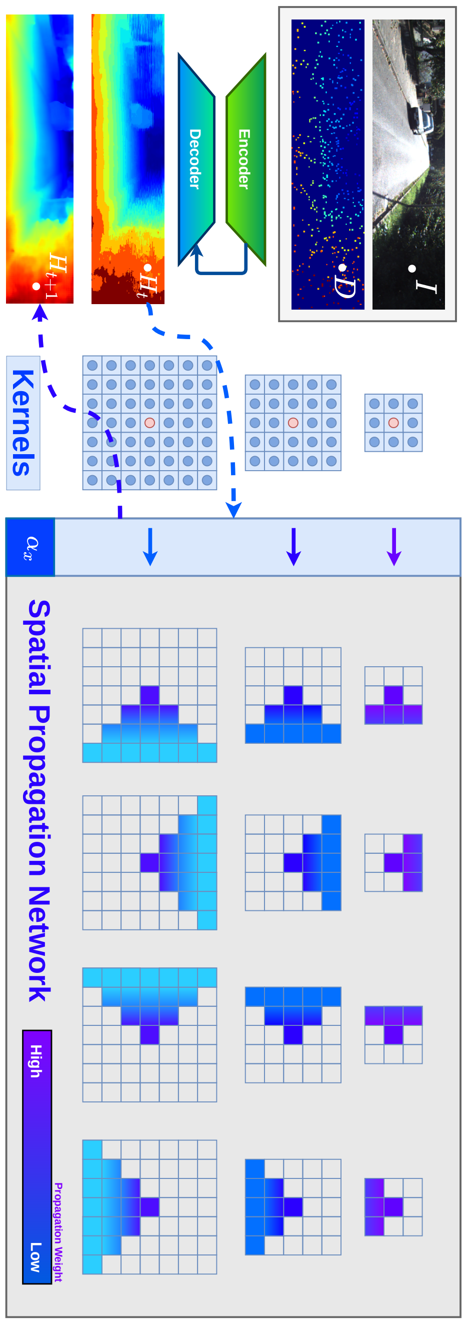


Figure 4.13: Sparse-to-dense depth estimation refinement. Each pixel (node) receives information from a 2 dimensional plane with three-way connection. (Best viewed in landscapes).

Fig. 4.13 represents the pipeline of the sparse-to-dense convolutional spatial propagation network. The RGB image I, along with the sparse depth map D is fed into the depth estimator with an encoder and decoder structure. The predicted hidden layer map contains dense depth information is further refined from a three-way Spatial Propagation Network [LDMG+17]. The Spatial Propagation Network is a process of diffusion evolution. But unlike a simple diffusion function, it also enables each pixel to connect to three pixels from the previous row/column, i.e., the left-top, middle and bottom pixels from the previous column for the left-to-right propagation direction, in practice, we propagate the depth map in four directions (as shown in bottom part of Fig. 4.13). The parallel four directional update scheme leads to significant performance improvement in both speed and quality over the serial ones.

By embedding the depth map $D \in \mathbf{R}^{m\times n}$ to hidden space $\mathbf{H}^{m\times n\times c}$ where c is the number of feature channels, we can diffuse the depth with the partial differential equation:

$$H^{t+1} - H^{t} = \frac{\partial H^{t+1}}{t} = -\alpha_x(I - A)H^{t} + (I + B)D^{0}. \tag{4.24}$$

Where α is the per-pixel estimation of the assembling parameter, A is the affinity matrix, and B is the degree matrix containing column-first vectorization of feature map [CWY18].

As mentioned above, pixels in hidden map H are locally connected throughout the propagation iterations. The connections indicate the pairwise similarities of pixels. The higher the similarity is, the farther the propagation is. Correspondingly, the propagation in areas with low similarity will be prevented. With this method, the iteration of the depth information will stop at the edge of the obstruction, thereby retaining the sharp outline of the object. Through the above operations, the refined depth map $H_t + 1$ retains fine details while keep the smoothness of the surfaces.

4.6 Gradient-based residual pattern

The pixel intensity of a small patch compared by direct visual odometry methods is extremely non-convex. This is a massive obstacle for second-order optimizers to converge into global or even local minima. So, in this section, we explore a method to constrain the photometric error to be as convex as possible. Actually, with a proper initialization (achieved by deep methods), the depth estimation can be very close to the ground truth.

We are not at a loss as to the non-convex nature of the image. Please remember that the subject we are studying is actually a set of images, that is, a video. In the case where the frame rate is large enough so that the deformation between adjacent images is small, the photometric error generated between the two images can almost be regarded as a convex function. This is also a prerequisite for most direct methods to function effectively. As we can see in the figure below (Fig. 4.14), when we project a point in the image to a position near the adjacent frame, the resulting residual partially forms an convex area.

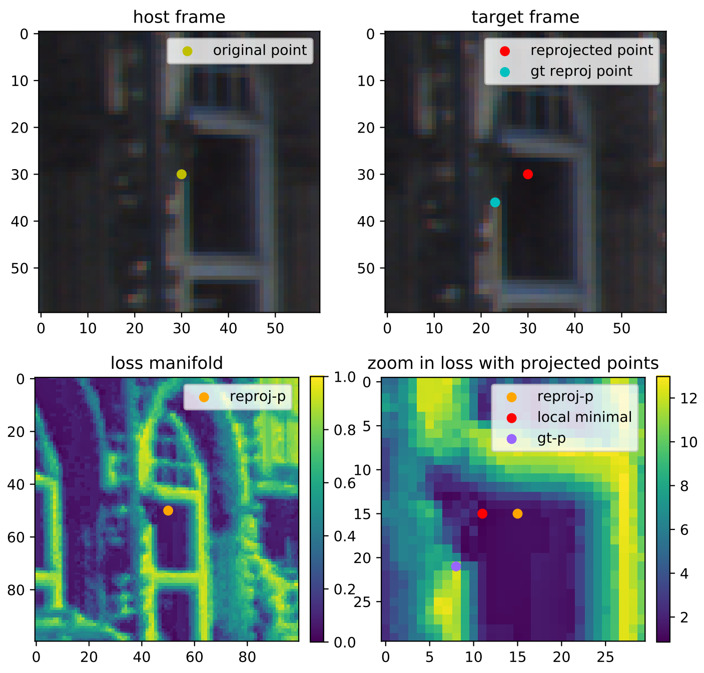


Figure 4.14: Photometric error manifold.

In prior work on monocular direct visual odometry, one only initializes the pose with the identity matrix [ZTS19] and uses inverse depth filter to iterate and refine the pose estimation. Since the depth and camera pose are coupled, the Hessian matrix will not be full-rank which leaves one degree of freedom in the nullspace [EKC17] and results in arbitrary initial scale of depth-pose pair. Over time, the lack of global constraints leads to error accumulation and drift in the scale of both pose and depth [YWGC18]. Although carefully calibrated cameras can secure a relatively constant scale within the local bundle adjustment window, the arbitrary scale initialized in the tracking process will no longer match with the previous one. This inability to recover scale across tracking failures severely reduces the robustness of monocular direct visual odometry.

Since DSO uses the Gauss-Newton method to solve bundle adjustment, it also requires good initialization and a locally convex photometric error. This forces the authors to select tracking points with high gradient as candidates. The high gradient points neighborhood will create a unique combination of pattern. The combination of these unique patterns enables the entire photometric error mainfold to approximate the convex function. However, Unlike previous methods, we use a relatively denser point cloud. The previous simple residual method cannot guarantee the local convex nature of photometric error. Since there are more points come from repeated textures or low texture image regions. The residuals in these places are basically the same, and the corresponding local photometric errors will also become flat. This is disastrous for gradient-dependent algorithms like Gauss-Newton method.

One of our strategies is to leverage those fixed map points as extra constraints on the optimization of the rest points. By doing this, we assume all points do not independently hold their own inverse depths. Instead, their inverse depths should be a likelihood that is defined by its neighborhood. We can thus propagate our likelihood from the known map points to those unknown points with the update of new observations (constraints). However, this method requires updating the dense map with many iterations until depth convergence, which is computationally expensive.

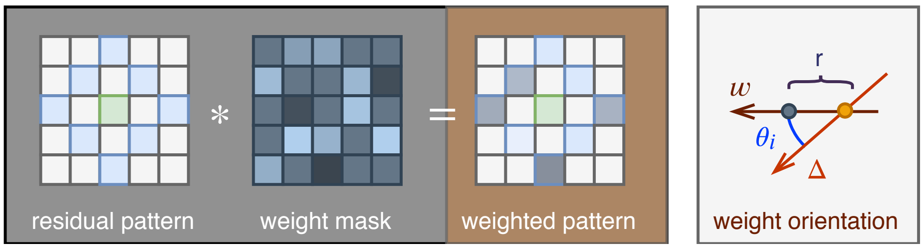


Figure 4.15: Weighted residual pattern.

Thus, we propose a more robust and simple method to capture the photometric error efficiently while keeping the local convexity: gradient-based dynamic residual pattern.

$$\phi(\theta_i, r_i) = \frac{cos^2\theta_i}{r_i}\frac{\partial I}{\partial x} + \frac{sin^2\theta_i}{r_i}\frac{\partial I}{\partial y}. \tag{4.25}$$

Where ϕ is the per pixel weight and it takes two parameters θ_i, and r_i, here i means each pixel. θ_i is the angle between current pattern pixel orientation w, and the reprojection center gradient orientation Δ, r is the corresponding offset distance. As shown in Fig. 4.16. The 9 cells of residual dimensions are weighted according to the neighbor gradient direction. After weighting, the residual pattern will look for points with similar gradient directions nearby instead of looking for all directions. When the relative motion between the two frames of image is small, the residual error obtained by reprojection can ensure local convexity, but when the relative motion is large, the reprojection area will be enlarged in the direction of motion, so that local convexity cannot be guaranteed. Adding gradient constraints in the process of obtaining residuals is equivalent to doing a simple corner matching locally. After matching, the photometric error manifold is smoother and convex, which improves the stability of the corresponding motion state.

The gradient orientation Δ is defined as following:

$$\Delta = tan^{-1}(\frac{\delta y}{\delta x}). \tag{4.26}$$

Where δy and δx are vertical and horizontal neighbour pixel difference. The process above mimicks orientation extraction in the SIFT feature [Low04], note that all residual patterns are weighted in different scale-spaces. This process benefits the low texture and low gradient regions since they assign more weights on the gradient direction and contribute more motion update correction in the same direction. When the tracking points are evenly sampled in the image space, gradients from all directions can offer different constraints on pose optimization.

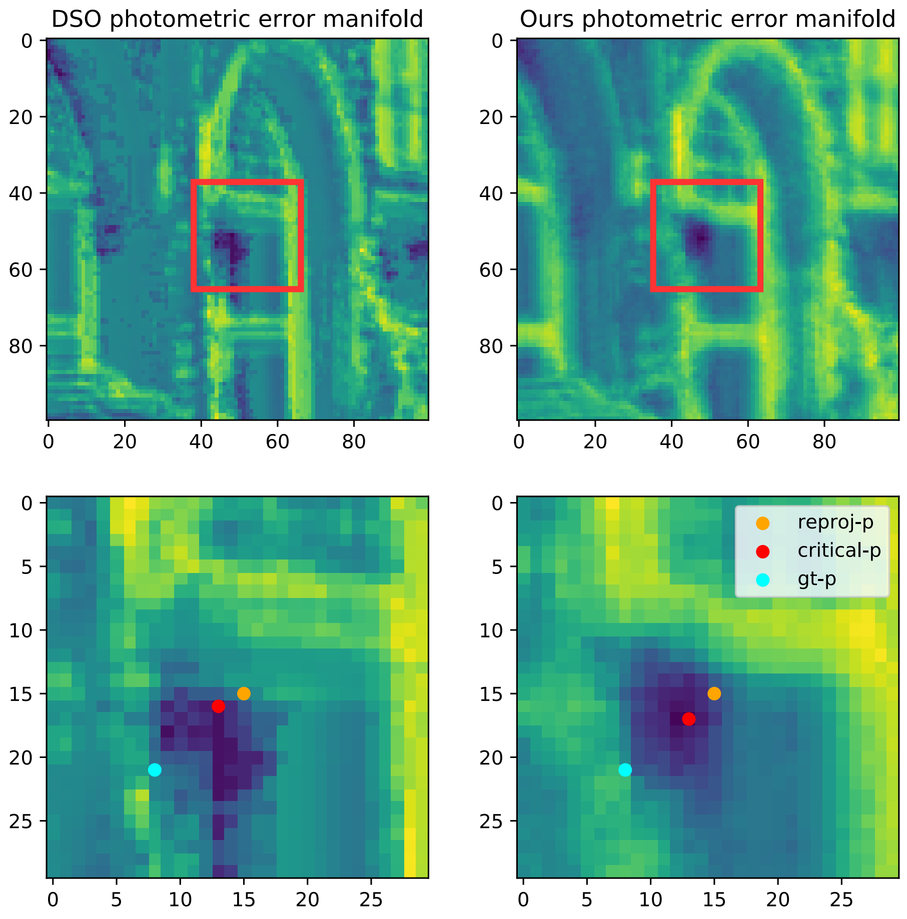


Figure 4.16: Comparison of photometric error generated from residual patterns of DSO and ours.

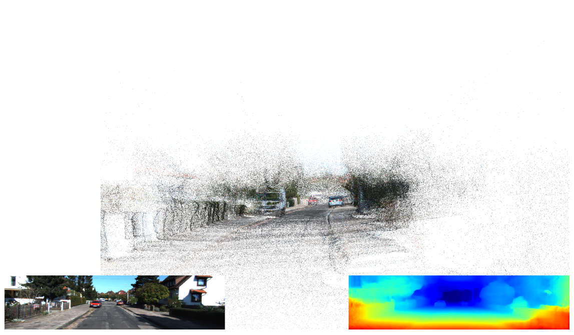
The weighted residual pattern can maintain a locally convex photometric error manifold (as shown in Fig. 4.16), which increase the convergence rate of the second-order optimization. The red dot in Fig. 4.16 is the local minimum within the photometric error patch. The left column in the figure is the photometric error manifold from a naive residual pattern, and the right column is the photometric error manifold with the gradient based residual pattern. We zoom in the error manifold in the second row of in the figure. In the figure, we can see that the photometric error generated by the weighting residual pattern is smoother and more convex than that of DSO. This convex optimization enables the Gauss-Newton algorithm that relies on gradient optimization to converge to a near-optimal minimum easier, therefore increases the tracking robustness and precision. Please note that due to the complexity of the actual optimization algorithm, we are unable to show the photometric error manifold generated by all points for different poses during the optimization process. We will discuss in detail the impact of the gradient based residual pattern proposed in this thesis on the localization results in an ablation study later.

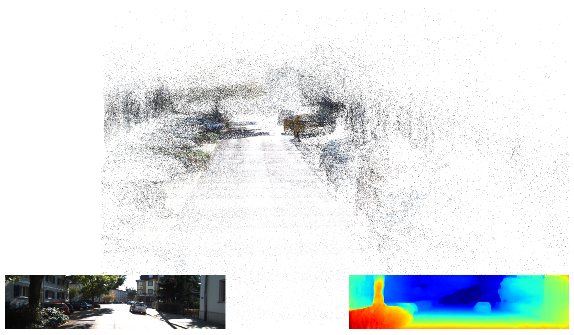
5 Experiments and analysis

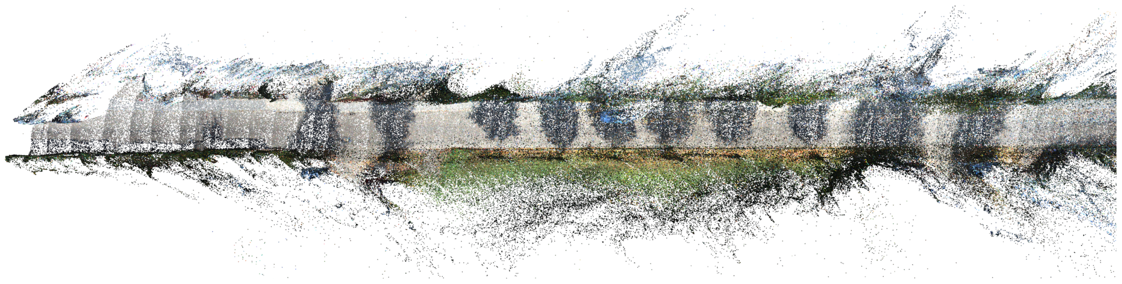
In this section, we demonstrate our experimental results evaluated on the odometry dataset of the KITTI Vision Benchmark [GLU12]. Our depth estimation and dense local bundle adjustment are performed on a Nvidia RTX 2080 max-Q GPU. Feature extraction and the loop closure (implemented with G^2O [KGS+11]) are all done by a CPU (Intel Core i7-8750H) running at 2.6 GHz with 16 GB RAM. The input image resolution is 1241 $\times$ 376. All images are pre-rectified. Depth inference takes additional time on the CPU (125 $\pm$ 50 ms), but can run faster on the GPU (20 $\pm$ 10 ms), which makes our GPU-based implementation run in real-time on a laptop computer. Next we will verify our algorithm from multiple angles, including qualitative results, quantitative results, ablation study and runtime.

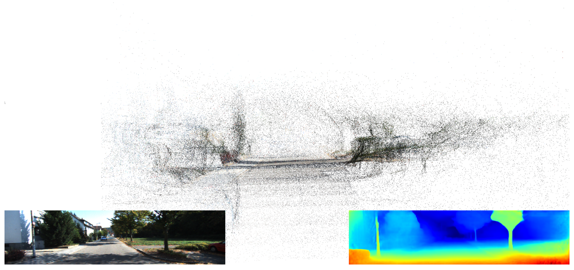
5.1 Qualitative and quantitative results

The following are the results of our qualitative experiments. We first showed the real-time results of the entire system on the test data set, and showed some renderings of 3D reconstruction. The following qualitative test results are taken from the KITTI dataset.


(a) Seq 03


(b) Seq 05


(c) Seq 19


(d) Seq 07

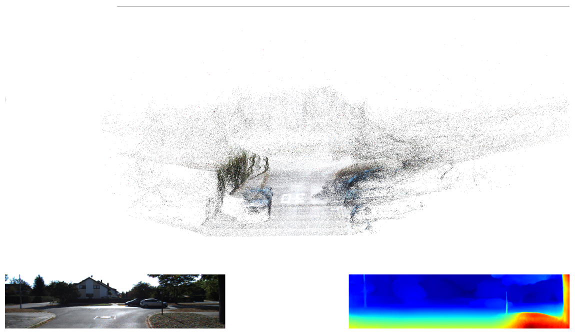

(e) Seq 12

Figure 5.1: Qualitative results taken from KITTI-Seq19.

Fig. 5.1 mainly reflects the results of 3D reconstruction and some localization results. The first and third row in Fig. 5.1 are captured in the live run of our system. The second row in Fig. 5.1 shows the bird eye view mapping result of a road in urban area. The complete localization results will be described in detail later. In the figure, we can find that adding the depth information of the image allows us to distribute the tracking points more evenly throughout the image, which is beneficial to improve the stability and accuracy of the localization. Our depth estimation perfectly solves the problem of scale drift. In the image we can see the flat ground and accurate vehicle model, which shows that our algorithm has achieved very satisfactory results.

Although we did not find a failure case of our algorithm on KITTI Dataset, our algorithm also inherited the limitation of some direct methods and failed in some special cases. If the exposure is abnormal or pure rotation, we still cannot effectively estimate the camera movement. The former is because the input image is invalid and there is no way to extract the tracking point. The latter is because the too large baseline causes the matching to fail.

As presented in Table 5.1, we evaluate our CNN-DVO method along with state-of-the-art direct visual odometry methods like DSO, LDSO, CNN-SVO and stereo DSO for comparison on all sequences from the KITTI Odometry training set (sample trajectories in Fig. 5.3 - Fig. 5.5). Since DSO and LDSO are initialized in arbitrary scale, we perform $\mathbf{Sim(3)}$ alignment with ground truth for evaluation consistency. In some experiments, we demonstrate better performance than Stereo DSO (sequences containing a lot of strong rotation motion). Since our point selection strategy more evenly samples points across the global image, the tracking thread is less affected by the frequent removal of outliers. Also, the fast depth initialization of new tracking points constrains the scale of the pose Hessians marginalization and regularizes the pose delta update close to the parameter tangent space.

We not only use depth images to enhance the visual odometer, but also use the localization results from visual odometer to update our depth image predictions. We project the 3D map points generated from visual odometer into a 2D image and convert the sparse depth image into a dense depth image via the sparse-to-dense method. This pipeline helps us refine our depth estimation, and improves both local details and global scales (as shown in Fig. 5.2).

Sequence		Absolute Pose Error (APE)					Relative Pose Error (RPE)				
	Method	Max	Mean	Min	Rmse	Std	Max	Mean	Min	Rmse	Std
seq 00	DSO	239.06	101.12	4.05	118.28	61.34	5.08	0.61	0.00	0.72	0.39
	SDSO	26.47	7.34	0.57	9.34	5.78	1.75	0.10	0.00	0.19	0.16
	LDSO	28.53	11.95	2.49	13.45	6.17	**0.87**	0.09	0.00	**0.11**	**0.06**
	CNN-SVO	64.41	17.66	4.07	20.11	9.63	2.26	0.09	0.00	0.15	0.12
	Ours	**14.66**	**3.57**	**0.10**	**4.55**	**2.83**	4.26	**0.08**	**0.00**	0.23	0.22
seq 01	DSO	33.82	**6.78**	1.24	**9.71**	6.95	22.25	0.18	0.02	0.88	0.86
	SDSO	41.55	7.05	1.99	9.79	**6.80**	**2.42**	**0.14**	**0.01**	0.28	0.24
	LDSO	29.25	7.65	**0.23**	10.48	7.16	2.81	0.15	0.01	**0.25**	**0.21**
	CNN-SVO	460.74	279.32	54.76	296.43	99.25	64.04	12.04	0.35	16.37	11.10
	Ours	**28.35**	8.00	0.33	9.85	7.91	3.02	0.16	0.03	0.33	0.36
seq 02	DSO	308.07	100.73	6.76	125.05	74.10	7.12	0.50	0.00	0.61	0.35
	SDSO	**11.62**	**5.22**	**0.32**	**5.81**	**2.56**	0.88	0.07	0.00	0.11	0.09
	LDSO	121.45	27.96	3.47	34.92	20.91	3.64	0.09	0.00	0.15	0.12
	CNN-SVO	84.03	45.80	21.85	48.25	15.17	7.39	0.58	0.01	0.74	0.47
	Ours	23.11	10.05	4.01	10.98	4.41	**0.50**	**0.05**	**0.00**	**0.06**	**0.04**
seq 03	DSO	6.90	1.97	**0.15**	2.33	1.24	1.02	0.05	0.01	**0.08**	**0.06**
	SDSO	**2.50**	**1.17**	0.27	**1.25**	**0.45**	**0.58**	**0.05**	**0.00**	0.09	0.07
	LDSO	8.53	2.57	0.17	2.99	1.54	1.12	0.06	0.00	0.09	0.07
	CNN-SVO	14.87	3.98	1.27	4.82	2.73	2.01	0.14	0.01	0.28	0.25
	Ours	12.98	5.53	1.72	6.27	2.95	0.60	0.16	0.04	0.19	0.10
seq 04	DSO	2.36	0.88	0.19	1.00	0.47	0.31	**0.03**	**0.00**	**0.04**	0.03
	SDSO	**1.03**	**0.55**	0.42	**0.56**	0.12	0.25	0.04	0.00	0.06	0.04
	LDSO	2.81	0.97	**0.13**	1.12	0.55	0.36	0.03	0.00	0.04	0.03
	CNN-SVO	6.36	2.35	1.31	2.50	0.86	2.00	0.30	0.03	0.42	0.30
	Ours	2.16	1.86	1.76	1.86	**0.07**	**0.12**	0.05	0.01	0.05	**0.02**
seq 05	DSO	174.91	55.78	10.95	65.11	33.58	3.25	0.46	0.00	0.57	0.34
	SDSO	26.61	7.46	**0.17**	8.97	4.98	2.87	0.10	0.00	0.21	0.19
	LDSO	13.53	3.80	1.04	4.25	1.90	0.58	**0.05**	**0.00**	**0.07**	**0.04**
	CNN-SVO	33.59	10.77	0.69	12.22	5.77	2.39	0.09	0.00	0.20	0.18
	Ours	**7.06**	**2.81**	1.00	**3.09**	**1.27**	**0.48**	0.06	0.00	0.08	0.05
seq 06	DSO	146.81	76.08	3.41	83.63	34.72	5.81	0.48	0.01	0.64	0.43
	SDSO	9.50	**4.29**	**0.60**	4.89	2.35	1.64	**0.08**	**0.00**	**0.14**	0.12
	LDSO	24.99	11.20	0.71	13.27	7.11	**0.58**	0.12	0.00	0.15	**0.09**
	CNN-SVO	220.98	71.30	22.59	81.73	39.95	6.18	0.89	0.09	1.15	0.73
	Ours	**7.58**	4.57	0.92	**4.82**	**1.52**	1.05	0.11	0.01	0.15	0.10
seq 07	DSO	64.96	19.33	0.36	23.43	13.23	0.98	0.25	0.01	0.32	0.19
	SDSO	4.16	2.18	0.39	2.35	0.89	0.95	0.08	0.00	0.13	0.10
	LDSO	47.98	17.58	4.52	20.10	9.74	2.43	0.20	0.01	0.27	0.19
	CNN-SVO	11.35	3.31	0.37	4.05	2.33	1.04	0.07	0.00	0.10	0.08
	Ours	**0.66**	**0.42**	**0.13**	**0.44**	**0.13**	**0.53**	**0.04**	**0.00**	**0.06**	**0.04**
seq 08	DSO	408.04	95.81	8.63	116.85	66.89	4.51	0.64	0.01	0.79	0.45
	SDSO	16.36	5.65	0.84	6.49	3.18	**1.46**	0.08	**0.00**	**0.15**	**0.13**
	LDSO	441.86	101.11	6.76	126.48	75.98	4.61	0.53	0.00	0.66	0.40
	CNN-SVO	30.39	8.73	0.99	10.65	6.10	2.43	0.11	0.00	0.17	0.13
	Ours	**14.08**	**3.81**	**0.22**	**4.51**	**2.42**	3.74	**0.07**	0.00	0.17	0.16
seq 09	DSO	164.56	57.58	3.11	68.21	36.57	11.24	0.32	0.00	0.48	0.36
	SDSO	**10.85**	**3.93**	**0.72**	**4.50**	**2.18**	1.27	0.09	0.00	0.15	0.12
	LDSO	165.46	64.18	4.45	75.80	40.34	1.25	0.31	0.01	0.37	0.20
	CNN-SVO	27.59	13.23	3.88	14.69	6.38	4.49	0.28	0.01	0.52	0.44
	Ours	13.85	4.26	1.49	4.90	2.42	**0.36**	**0.05**	**0.00**	**0.06**	**0.04**
seq 10	DSO	44.07	11.27	2.65	13.68	7.76	0.75	0.14	0.00	0.19	0.13
	SDSO	3.19	**1.08**	**0.19**	**1.20**	0.53	0.40	**0.03**	**0.00**	0.06	0.05
	LDSO	49.29	14.83	3.67	17.68	9.63	0.43	0.15	0.00	0.19	0.12
	CNN-SVO	25.93	7.53	1.78	8.74	4.44	3.02	0.29	0.01	0.45	0.35
	Ours	**2.46**	1.44	0.71	1.49	**0.41**	**0.27**	0.04	0.00	**0.05**	**0.04**
seq 10	DSO	44.07	11.27	2.65	13.68	7.76	0.75	0.14	0.00	0.19	0.13
	SDSO	3.19	**1.08**	**0.19**	**1.20**	0.53	0.40	**0.03**	**0.00**	0.06	0.05
	LDSO	49.29	14.83	3.67	17.68	9.63	0.43	0.15	0.00	0.19	0.12
	CNN-SVO	25.93	7.53	1.78	8.74	4.44	3.02	0.29	0.01	0.45	0.35
	Ours	**2.46**	1.44	0.71	1.49	**0.41**	**0.27**	0.04	0.00	**0.05**	**0.04**
Avg†	DSO	144.87	47.94	3.77	57.03	30.62	5.67	0.33	0.01	0.48	0.33
	SDSO	13.99	**4.17**	**0.59**	5.01	2.71	**1.32**	**0.08**	**0.00**	0.14	0.12
	LDSO	84.88	23.98	2.51	29.14	16.46	1.70	0.16	0.00	0.21	0.14
	CNN-SVO	89.11	42.18	10.32	45.84	17.51	8.84	1.35	0.05	1.87	1.29
	Ours	**11.54**	4.21	1.13	**4.80**	**2.39**	1.36	0.08	0.01	**0.13**	**0.11**
Min†	DSO	2.36	0.88	0.15	1.00	0.47	0.31	**0.03**	**0.00**	**0.04**	0.03
	SDSO	1.03	0.55	0.17	0.56	0.12	0.25	0.03	0.00	0.06	0.04
	LDSO	2.81	0.97	0.13	1.12	0.55	0.36	0.03	0.00	0.04	0.03
	CNN-SVO	6.36	2.35	0.37	2.50	0.86	1.04	0.07	0.00	0.10	0.08
	Ours	**0.66**	**0.42**	**0.10**	**0.44**	**0.07**	**0.12**	0.04	0.00	0.05	**0.02**

Table 5.1: Quantitative results of trajectory APE (m) and RPE (m) evaluated on KITTI dataset. Avg† and Min† are average and min metrics over all tested sequences. **Bold** numbers have the smallest error.

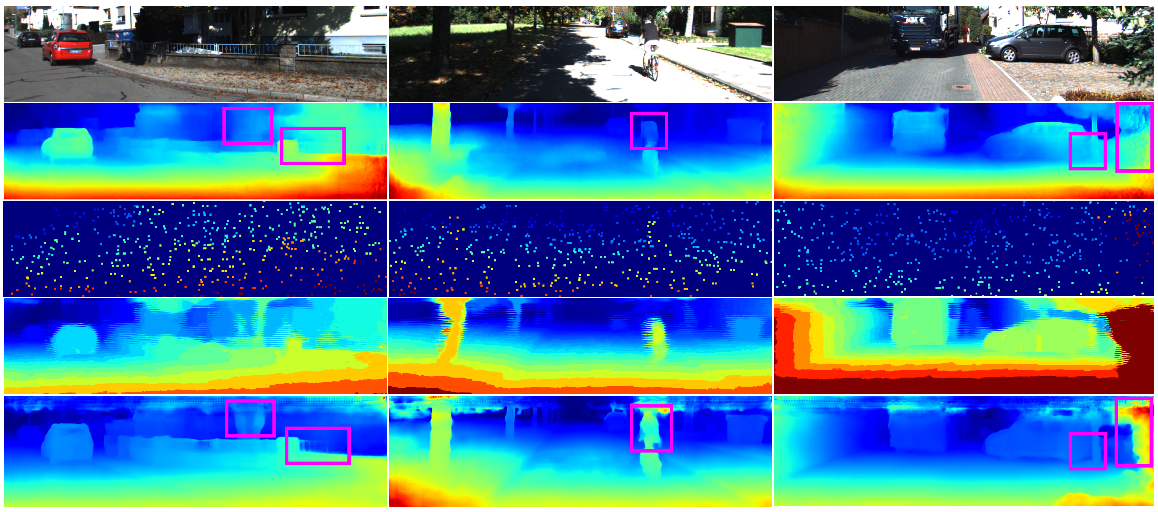


Figure 5.2: Qualitative results, sparse-to-dense refinement.

The first row in Fig. 5.2 is the original image, the second row is the original depth estimate (from MonoDepth2 [GMAFB19]), and the third row is the sparse map points, which are estimated by the previous visual odometer. The fourth and fifth row are the results of sparse-to-dense matching, where the fourth line is the intermediate result of 10 iterations, and the last row is the final result. From the last row and second row of the figure (highlighted with purple rectangles), we can see that the depth image we estimated with sparse-to-dense has a corresponding improvement in both detail and overall level compared to the original depth image.

Our sparse-to-dense refinement are compared to existing methods on KITTI 2015 Eigen Split dataset [GLU12]. From Table 5.2, we can see that our depth refinement achieved second-best on the dataset, and does so by a large performance margin compared to the current state-of-the-art methods other than CSPN [CWGY19]. Also, it demonstrates a significant improvement in the depth refinement compared to the coarse depth predictions directly inferred from our deep CNN backbone (MonoDepth2 [GMAFB19]). Note that all results here are presented without post-processing, For a fair comparison, we cropped all the depth estimation output images into the same shape, which aligns with our sparse to dense recover output. Since the successful tracked map points are dropped from the upper part of the image.

Method	Config	Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Eigen [EPF14]	D	0.203	1.548	6.307	0.282	0.702	0.890	0.890
Liu [LSLR15]	D	0.201	1.584	6.471	0.273	0.680	0.898	0.967
Klodt [KV18]	DM	0.166	1.490	5.998	-	0.778	0.919	0.966
AdaDepth [NKKUPVB18]	D	0.167	1.257	5.578	0.237	0.771	0.922	0.971
MonoDepth2 [GMAFB19]	DS	0.115	0.903	4.863	0.193	0.877	0.959	0.981
Kuznietsov [KSL17]	DS	0.113	0.741	4.621	0.189	0.862	0.960	0.986
DVSO [YWSC18]	DS	0.097	0.734	4.442	0.117	0.891	0.965	0.984
CSPN [CWY18]	SDM	**0.015**	**0.135**	**1.129**	**0.112**	0.313	0.105	**0.001**
Ours	SDM	0.023	0.152	1.169	0.157	**0.270**	**0.086**	0.024

D: Depth supervision.
M: Self-supervised mono supervision.
S: Self-supervised stereo supervision.
Best results in each category are in bold.

Table 5.2: Comparison results on KITTI dataset between ours and other state-of-the-art depth estimation strategies.

We use the Eigen Split HDF5 [EPF14] data to prepare our training set and validation set. For monocular self supervised training which uses monocular sequences, we follow static frame removal method of Zhou *et al.*, [ZBSL17]. This results in 39,810 monocular images for training and 4,424 for validation. We use the same intrinsic for all images (image shape are unified through cropping), setting the principal point of the camera to the image center and the focal length to the average of all the focal lengths in KITTI.

5.2 Ablation study

Using the KITTI Odometry Benchmark, we show the effect of each design decision of our system. Table 5.3 contains the results of our approach without incorporating depth priors into the pipeline, without integrating loop closure for pose optimization, and while using a predicted pose prior for initialization. The evaluation results are Absolute Trajectory Error (m) on all KITTI training sequences. Experiments are averaged over 7 runs of each sequence.

Table 5.3: Ablation study of major components.

Seq	Ours	Ours $w\backslash o$ DP*	Ours $w\backslash o$ LC*	Ours $w\backslash$ PP*	Ours $w\backslash o$ RP*	CNN-SVO
00	**4.55**	118.28	13.45	9.34	6.73	130.03
01	9.79	9.71	10.48	**6.79**	12.04	202.36
02	10.98	125.05	34.92	5.81	**4.17**	48.24
03	**1.05**	2.33	2.99	1.25	2.08	3.26
04	0.52	0.98	1.12	**0.38**	0.73	2.10
05	2.23	65.11	4.25	8.97	**1.79**	20.39
06	**3.04**	83.63	13.27	4.89	6.16	12.50
07	**0.44**	23.42	2.10	2.35	3.99	4.05
08	**4.51**	116.85	6.48	6.49	7.19	10.65
09	4.90	68.21	5.80	**4.50**	4.90	14.69
10	1.49	13.68	7.68	**1.20**	1.63	8.74
mean	**3.96**	57.02	9.14	**5.00**	**5.49**	41.55

$w\backslash o$ DP*: without depth prior ($\mathbf{Sim}(3)$ scale aligned).
$w\backslash o$ LC*: without loop closure for pose optimization.
$w\backslash$ PP*: with pose prior for initialization.
$w\backslash o$ RP*: without gradient based residual pattern.
Evaluated with Absolute Trajectory Error (m). **Bold** numbers have the smallest error.

Starting with our principle contribution, our experiments show that the inclusion of depth priors helps to progressively reduce scale-drift over long sequences by restricting the reprojection of tracking points in the local bundle adjustment near the global minimum. The remaining accumulated estimation errors can then be further corrected with pose optimization through loop closure. We find that initializing our system with a predicted pose prior provides a relatively small accuracy boost on select sequences, but on average, does not improve the performance of the system. We can also find that the system tracking performance drops without the gradient based residual pattern.

In Fig 5.6, 5.7, 5.8 and 5.9 we show our results on the testing set. We show the same plots as recommended by the benchmark, where translational errors and rotational errors with respect to different distance intervals over the entire set are plotted. We have demonstrate that our method performs well in comparison to all other methods on the listed odometry sequences.

5.3 Runtime evaluations

Since we estimate the depth in low-resolution, it takes average 50 ms for each frame. A significant amount of time is also saved by running the system with fewer tracking points (increased sparsity). The overall runtime is slightly slower than DSO, but the initialization process is faster.

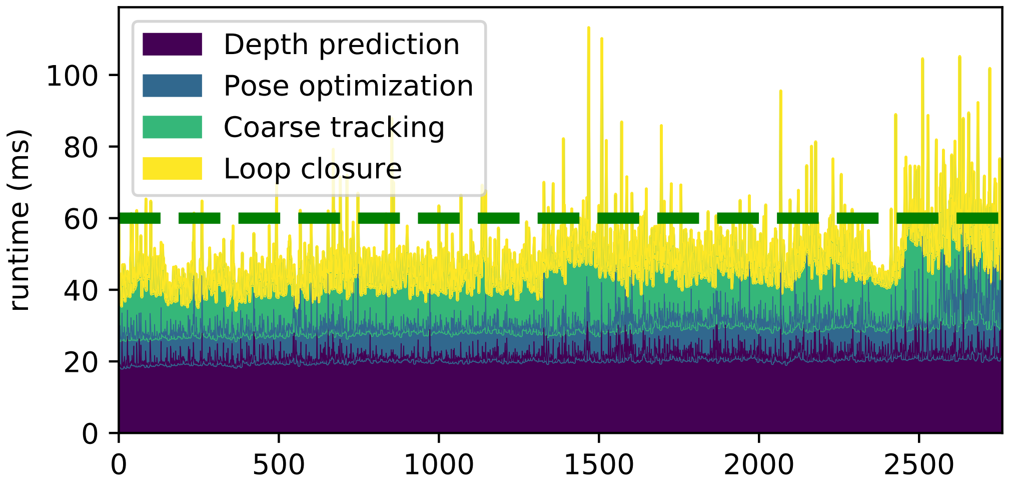


Figure 5.10: Run time evaluation on KITTI dataset seq 05. Execution time are collected for all components in our system for each frame. We take the average time result of 7 experiments.

Fig. 5.10 shows the processing time for each frame required when we run our complete system on sequence 05 of the KITTI Vision Benchmark. Note that the inference for depth priors are done on our GPU; the average inference time is 23 ms.

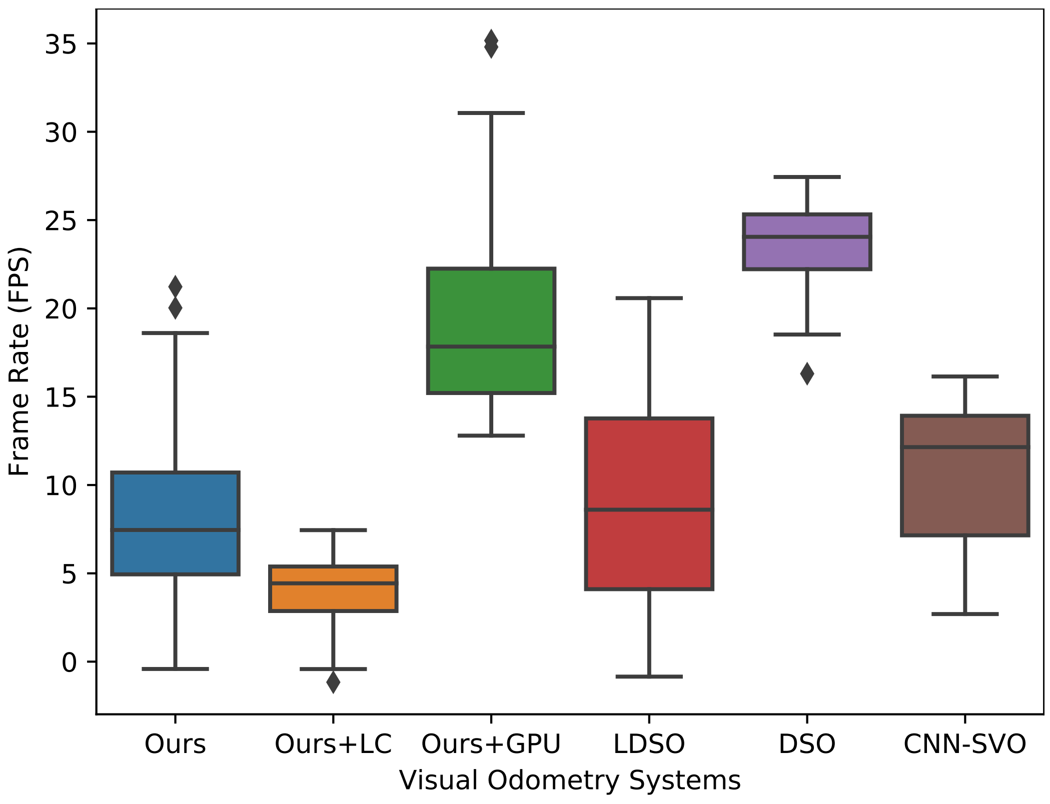
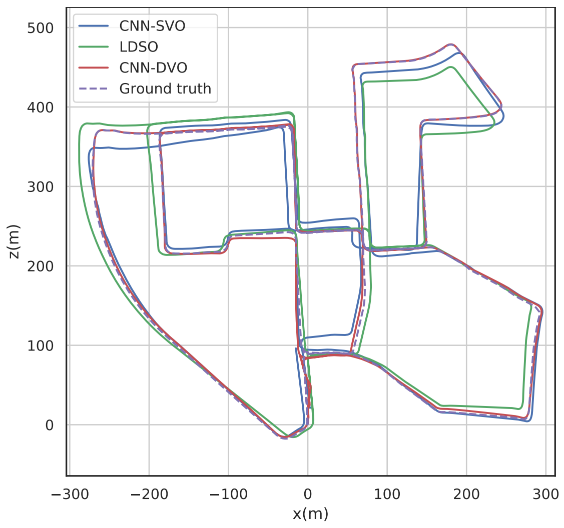
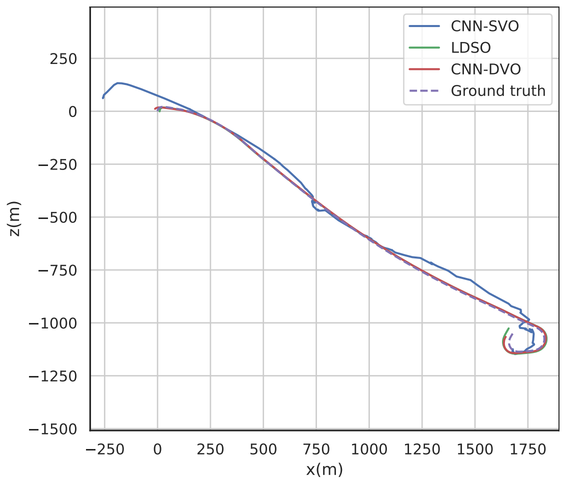


Figure 5.11: Comparison of visual odometry system frame rate results collected on all KITTI odometry. Note all experiments are performed on the same hardware and system configuration.

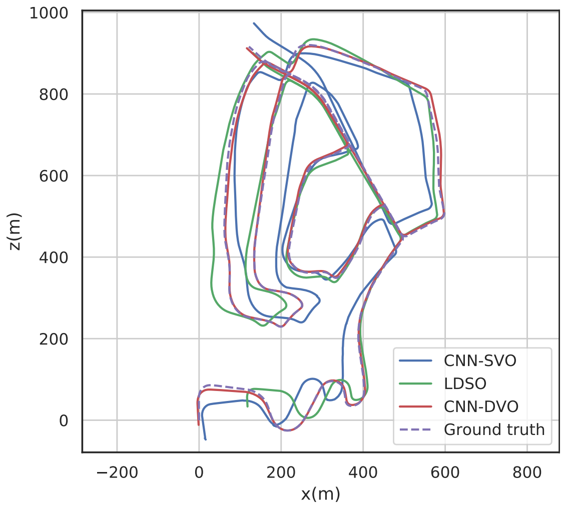
As shown in Fig. 5.11, our tracking pipeline operates at $\sim$15 FPS, while LDSO runs at $\sim$13 FPS, DSO at $\sim$28 FPS, and CNN-SVO at $\sim$14 FPS. For the front-end pose update with loop closure, we operate at $\sim$3 FPS, with LDSO slightly quicker at $\sim$4 FPS. We collect all the frame rate data of visual odometry systems listed in Fig. 5.11 running over all KITTI odometry sequences. All the experiments are done one the same configuration and hardware mentioned in the beginning of Chapter 5.



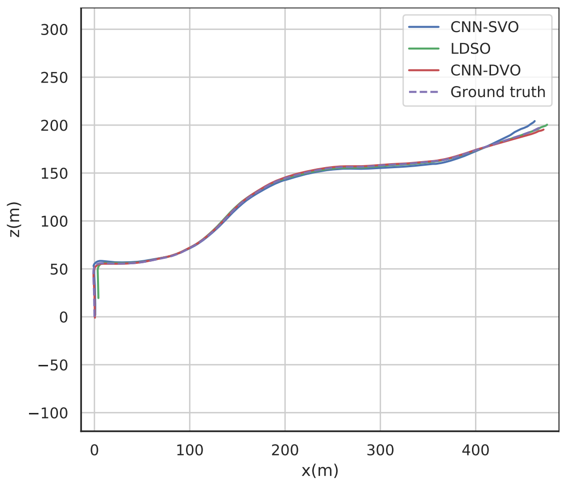
(a) Seq 00



(b) Seq 01

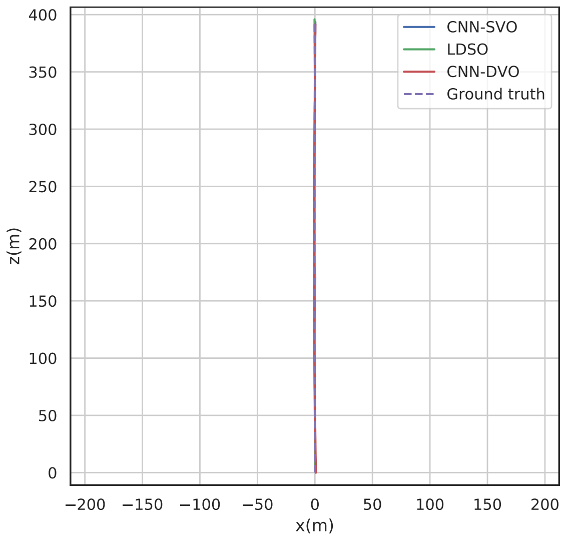


(c) Seq 02

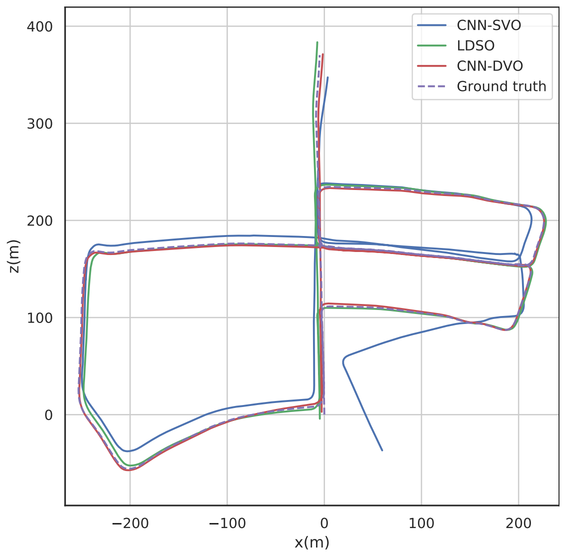


(d) Seq 03

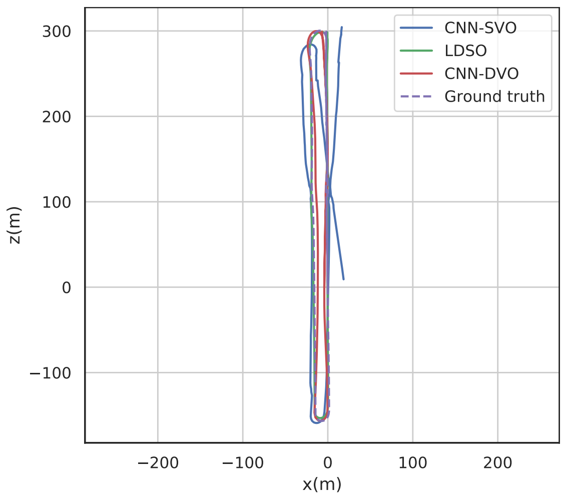
Figure 5.3: Trajectory comparison in KITTI dataset. Part I of figure (to be continued below).



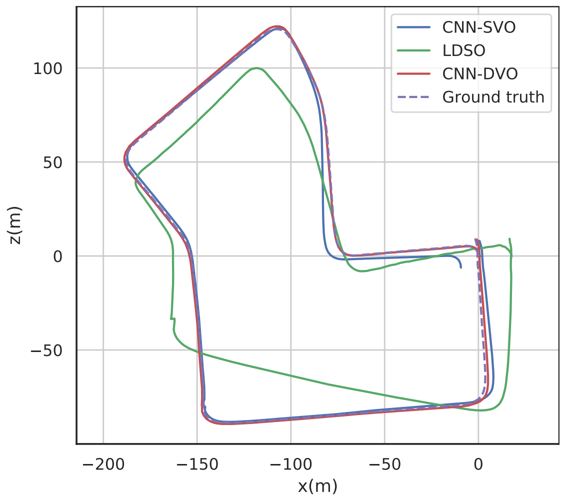
(a) Seq 04



(b) Seq 05

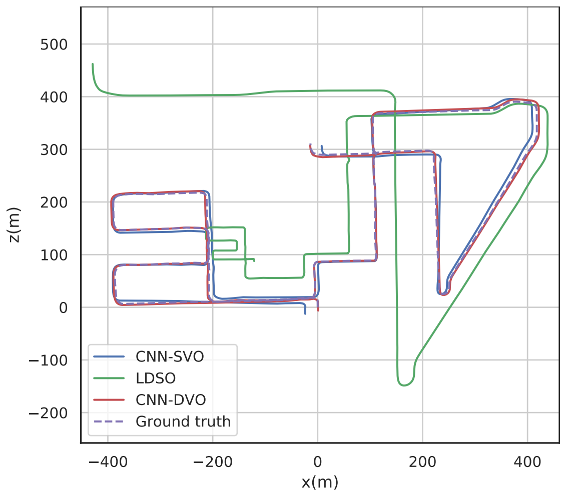


(c) Seq 06

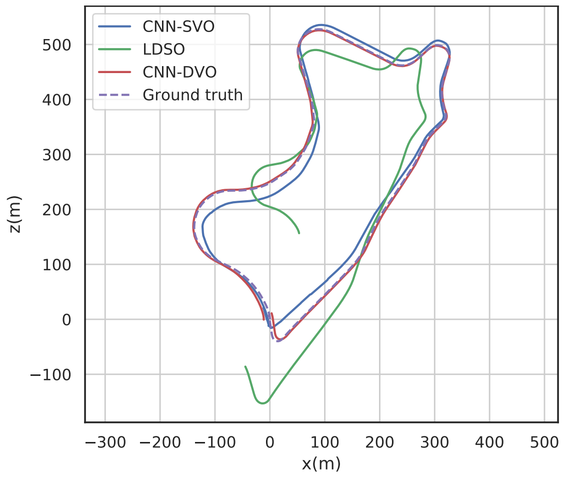


(d) Seq 07

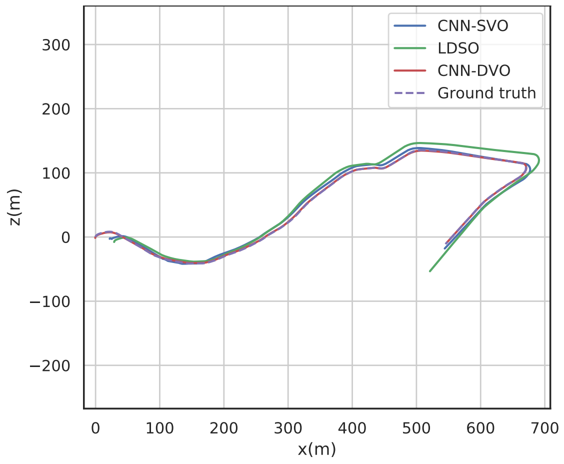
Figure 5.4: Trajectory comparison in KITTI dataset. Part II of figure (to be continued below).



(a) Seq 08

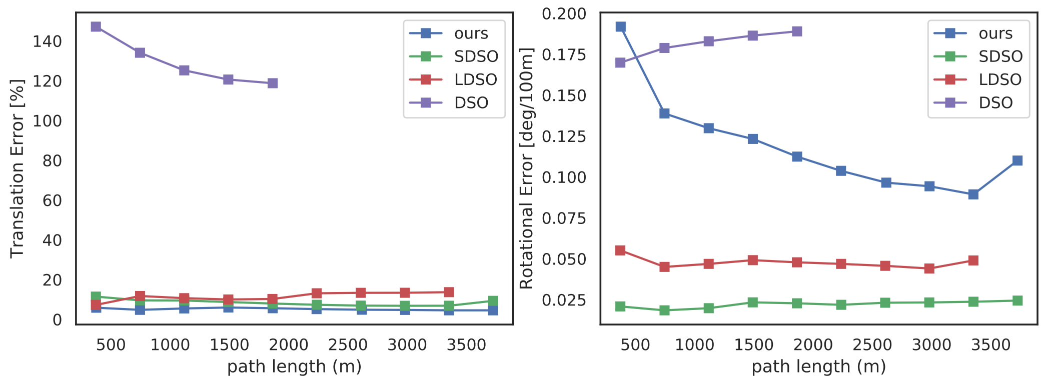


(b) Seq 09

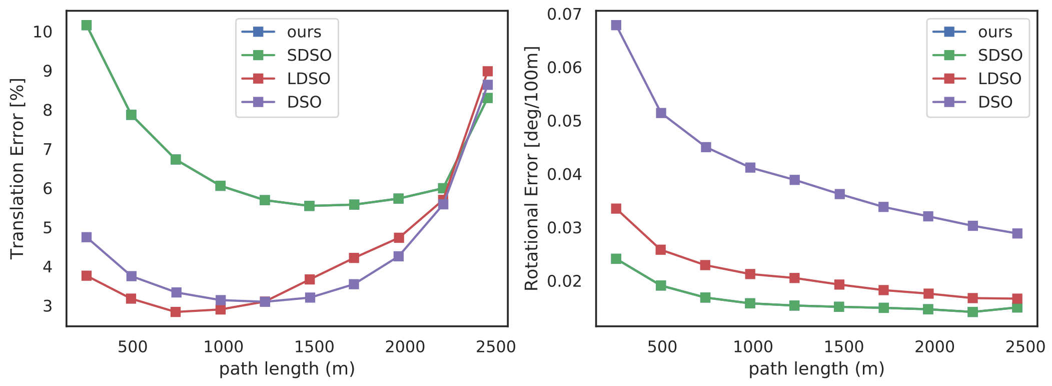


(c) Seq 10

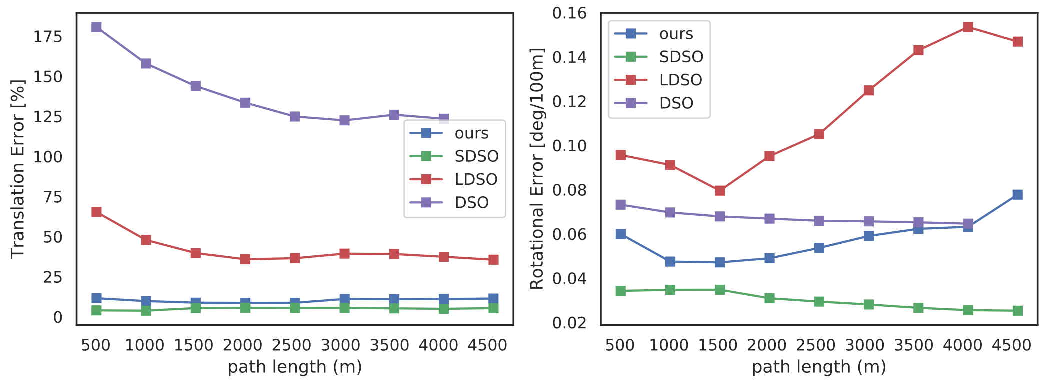
Figure 5.5: Trajectory comparison in KITTI dataset. Part III of figure.



(a) Seq 00

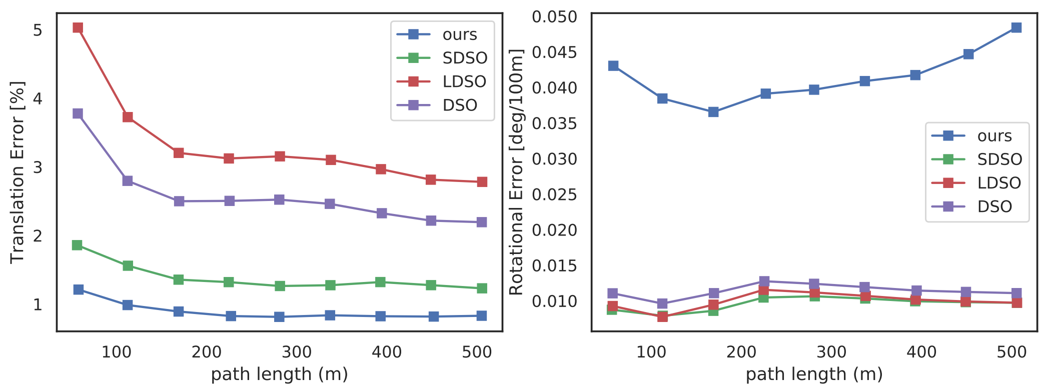


(b) Seq 01

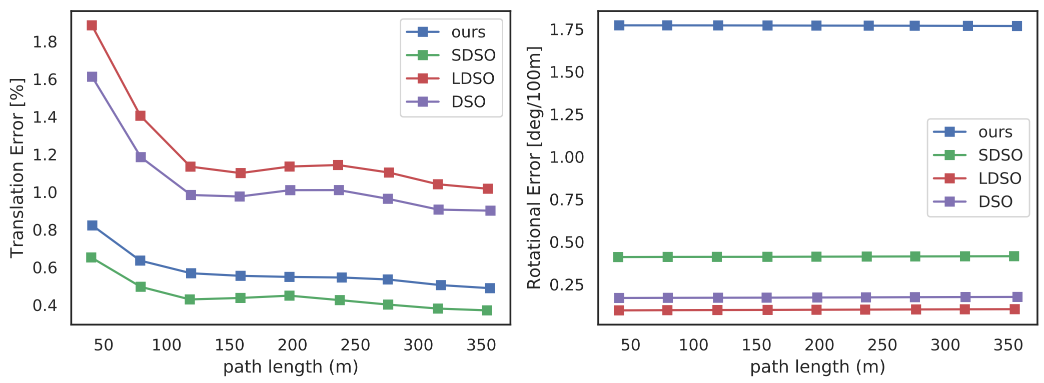


(c) Seq 02

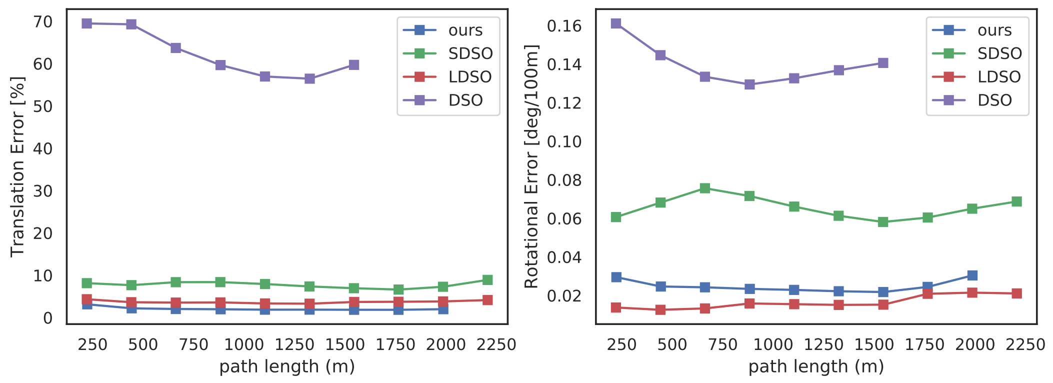
Figure 5.6: Average translational and rotational errors with respect to driving intervals on KITTI odometry dataset (Seq. 00-10). In most of the cases, our VO results (without loop closure) have almost the same accuracy as LDSO (with loop closure). We apply $Sim(3)$ Alignment on the entire trajectory for all tested methods, so LDSO performs best in some cases. Part **I** of figure (to be continued below).



(a) Seq 03

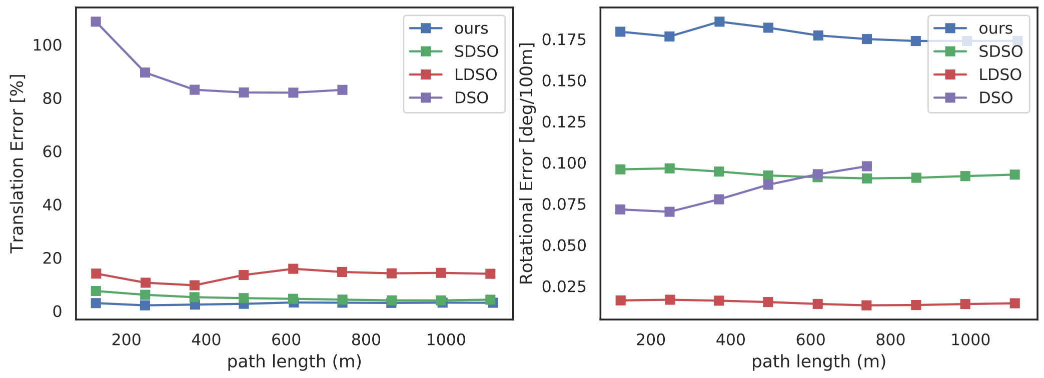


(b) Seq 04

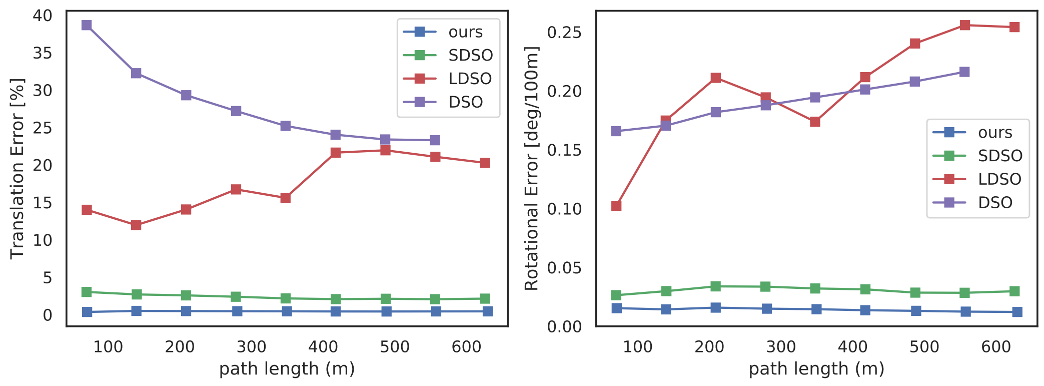


(c) Seq 05

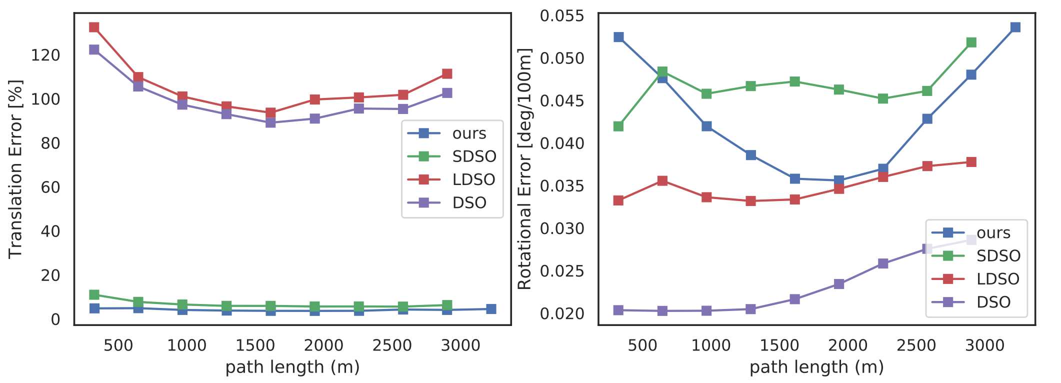
Figure 5.7: Average translational and rotational errors with respect to driving intervals on KITTI odometry dataset (Seq. 00-10). In most of the cases, our VO results (without loop closure) have almost the same accuracy as LDSO (with loop closure). We apply $Sim(3)$ Alignment on the entire trajectory for all tested methods, so LDSO performs best in some cases. Part **II** of figure (to be continued below).



(a) Seq 06

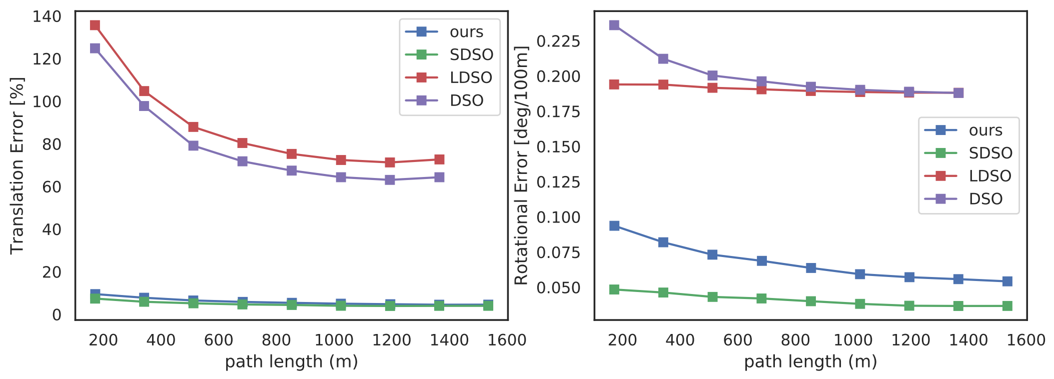


(b) Seq 07

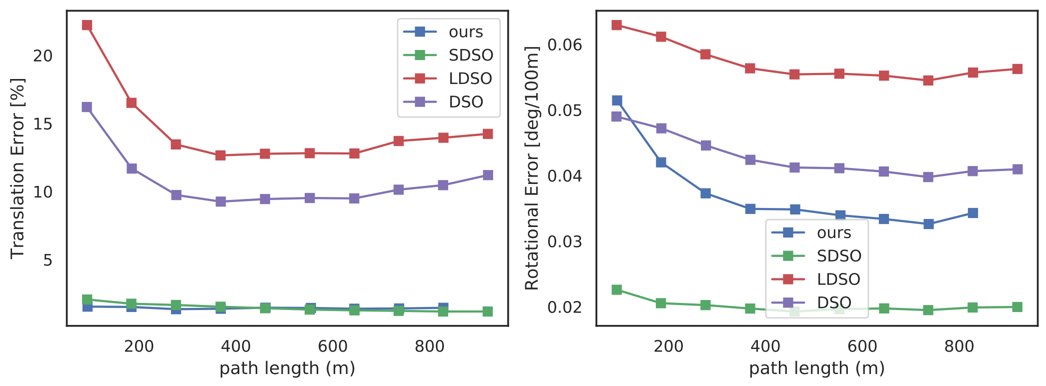


(c) Seq 08

Figure 5.8: Average translational and rotational errors with respect to driving intervals on KITTI odometry dataset (Seq. 00-10). In most of the cases, our VO results (without loop closure) have almost the same accuracy as LDSO (with loop closure). We apply $Sim(3)$ Alignment on the entire trajectory for all tested methods, so LDSO performs best in some cases. Part **III** of figure (to be continued below).



(a) Seq 09



(b) Seq 10

Figure 5.9: Average translational and rotational errors with respect to driving intervals on KITTI odometry dataset (Seq. 00-10), In most of the cases, our VO results (without loop closure) have almost the same accuracy as LDSO (with loop closure). We apply $Sim(3)$ Alignment on the entire trajectory for all tested methods, so LDSO performs best in some cases. Part **IV** of figure.

6 Final Conclusion & Future Work

In this thesis, we present a novel deep learning powered direct visual odometry system. To our knowledge, it is the only direct formulation that fully integrates depth prediction with every major system component (initialization, tracking, marginalization) to jointly optimize for all model parameters including inverse depth, camera pose and affine model in real-time (with GPU-based implementation). Extra constraints given by the depth prior provide strong benefits to our system, as it **a)** fixes the arbitrary initialization scale into constant scale, **b)** it greatly restrains the scale-drift during tracking, and **c)** it recovers the 3D correspondence of the measurements in loop closure while helping to maintain consistent scale. We also propose an improved point selection strategy to sample points near both low and high gradient image regions to avoid clustering, rendering our tracking thread more adaptable to small baseline stereo problems (e.g. in-place rotation). We have demonstrated that our approach achieves state-of-the-art results on the KITTI Odometry Benchmark [GLU12].

We also extended our pipeline to include fine-tuning of the deep CNN depth predictor. This fine-tuning process is achieved by employing a sparse-to-dense network to reconstruct a dense depth map from the newly optimized sparse depths after local bundle adjustment. To this end, we bridged the current optimization framework with depth refinement to maintain precise pose and depth estimations in new and challenging domains. Through the proposed sparse-to-dense depth refinement network, we have further improved the accuracy of depth prediction in terms of overall scale and local details. We obtained the state-of-the-art

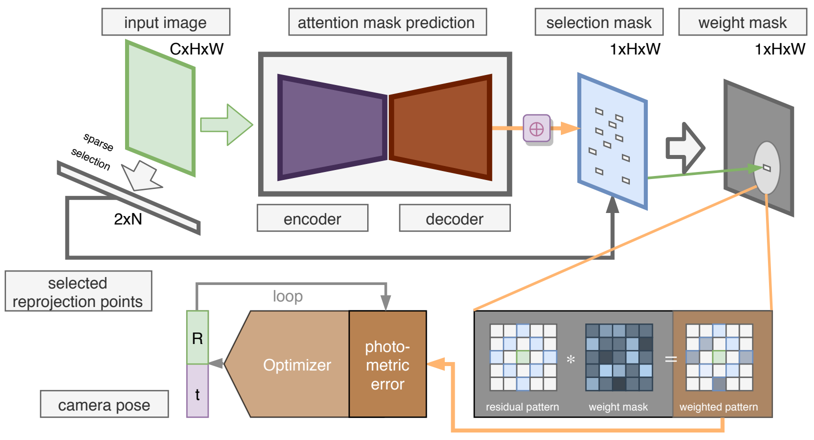


Figure 6.1: Unsupervised residual pattern learning pipeline.

prediction results on this public data set.

We aim to further extend this work in two main parts. One part is to loop the depth refinement back into the depth prediction networks to offer better ground truth level depth labels. Typically outdoor depth ground truth is very hard to obtain. Filling this gap will benefit most monocular depth estimators, and visual odometry methods. Another part is trying to further explore the possibility of leveraging the convolutional neural network to help learn a function to isolate the photometric error from direct inclusion of images (as shown in Fig. 6.1).

This work is part of a growing body of results that support the development of fully autonomous vision-based navigation. We hope we have contributed to a rapidly advancing front of research that will make real time autonomy a practical reality.

List of Publications

- Depth Estimation for Monocular Direct Visual Odometry. Ran Cheng, Christopher Agia, David Meger, Gregory Dudek. **CRV 2020**
- Vision-Based Autonomous Underwater Swimming in Dense Coral for Combined Collision Avoidance and Target Selection. Travis Manderson, Juan Camilo Gamboa Higuera, Ran Cheng, Gregory Dudek. **IROS 2018**
- Navigation in the Service of Enhanced Pose Estimation. Travis Manderson, Ran Cheng, David Meger, Gregory Dudek. **ISER 2018**

Bibliography

[Adi85] Gilad Adiv. Determining three-dimensional motion and structure from optical flow generated by several moving objects. *IEEE transactions on pattern analysis and machine intelligence*, (4):384–401, 1985.

[AHB37] KS Arun, TS Huang, and SD Blostein. Least-squares fitting of two 3-d point sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 698–700, 1937.

[AMSI16] Mohammad OA Aqel, Mohammad H Marhaban, M Iqbal Saripan, and Napsiah Bt Ismail. Review of visual odometry: types, approaches, challenges, and applications. *SpringerPlus*, 5(1):1897, 2016.

[ASdG$^+$19] Yasin Almalioglu, Muhamad Risqi U Saputra, Pedro PB de Gusmao, Andrew Markham, and Niki Trigoni. Ganvo: Unsupervised deep monocular visual odometry and depth estimation with generative adversarial networks. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 5474–5480. IEEE, 2019.

[Bak84] Robin Baker. *Bird navigation: the solution of a mystery?* CUP Archive, 1984.

[BC86] Ted J Broida and Rama Chellappa. Estimation of object motion parameters from noisy images. *IEEE transactions on pattern analysis and machine intelligence*, (1):90–99, 1986.

[BOHS15] Michael Bloesch, Sammy Omari, Marco Hutter, and Roland Siegwart. Robust visual inertial odometry using a direct ekf-based approach. In *2015 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 298–304. IEEE, 2015.

[BWC17] Paul Bergmann, Rui Wang, and Daniel Cremers. Online photometric calibration of auto exposure video for realtime visual odometry and slam. *IEEE Robotics and Automation Letters*, 3(2):627–634, 2017.

[CAMD20] Ran Cheng, Christopher Agia, David Meger, and Gregory Dudek. Depth estimation for direct visual odometry. In *preprint manuscript*, 2020.

[CCCK16] Hsiang-Jen Chien, Chen-Chi Chuang, Chia-Yen Chen, and Reinhard Klette. When to use what feature? sift, surf, orb, or a-kaze features for monocular visual odometry. In *2016 International Conference on Image and Vision Computing New Zealand (IVCNZ)*, pages 1–6. IEEE, 2016.

[CDM08] Javier Civera, Andrew J Davison, and JM Martinez Montiel. Inverse depth parametrization for monocular slam. *IEEE transactions on robotics*, 24(5):932–945, 2008.

[Ced13] Judith N Cederberg. *A course in modern geometries*. Springer Science & Business Media, 2013.

[Clo78] William F Clocksin. Determining the orientation of surfaces from optical flow. In *Proceedings of the 1978 AISB/GI Conference on Artificial Intelligence*, pages 93–102, 1978.

[CMR07] Andrew I Comport, Ezio Malis, and Patrick Rives. Accurate quadrifocal tracking for robust 3d visual odometry. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 40–45. IEEE, 2007.

[CWGY19] Xinjing Cheng, Peng Wang, Chenye Guan, and Ruigang Yang. CSPN++: Learning Context and Resource Aware Convolutional Spatial Propagation Networks for Depth Completion. *arXiv e-prints*, page arXiv:1911.05377, Nov 2019.

[CWY18] Xinjing Cheng, Peng Wang, and Ruigang Yang. Learning depth with convolutional spatial propagation network. *arXiv preprint arXiv:1810.02695*, 2018.

[dlEIM$^+$16] Arturo de la Escalera, Ebroul Izquierdo, David Martín, Basam Musleh, Fernando García, and José María Armingol. Stereo visual odometry in urban environments based on detecting ground features. *Robotics and Autonomous Systems*, 80:1–10, 2016.

[DM02] Andrew J Davison and David W Murray. Simultaneous localization and map-building using active vision. *IEEE transactions on pattern analysis and machine intelligence*, 24(7):865–880, 2002.

[DRMS07] Andrew J Davison, Ian D Reid, Nicholas D Molton, and Olivier Stasse. Monoslam: Real-time single camera slam. *IEEE transactions on pattern analysis and machine intelligence*, 29(6):1052–1067, 2007.

[EKC17] Jakob Engel, Vladlen Koltun, and Daniel Cremers. Direct sparse odometry. *IEEE transactions on pattern analysis and machine intelligence*, 40(3):611–625, 2017.

[EKC18] J. Engel, V. Koltun, and D. Cremers. Direct sparse odometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(3):611–625, March 2018.

[EPF14] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. In *Advances in neural information processing systems*, pages 2366–2374, 2014.

[ESC13] Jakob Engel, Jurgen Sturm, and Daniel Cremers. Semi-dense visual odometry for a monocular camera. In *Proceedings of the IEEE international conference on computer vision*, pages 1449–1456, 2013.

[ESC14] Jakob Engel, Thomas Schöps, and Daniel Cremers. Lsd-slam: Large-scale direct monocular slam. In *European conference on computer vision*, pages 834–849. Springer, 2014.

[Ett01] Scott M Ettinger. Design and implementation of autonomous vision-guided micro air vehicles. Master's thesis, University of Florida, 2001.

[FCDS16] Christian Forster, Luca Carlone, Frank Dellaert, and Davide Scaramuzza. On-manifold preintegration for real-time visual–inertial odometry. *IEEE Transactions on Robotics*, 33(1):1–21, 2016.

[FPRARM15] Jorge Fuentes-Pacheco, José Ruiz-Ascencio, and Juan Manuel Rendón-Mancha. Visual simultaneous localization and mapping: a survey. *Artificial intelligence review*, 43(1):55–81, 2015.

[FPS14] Christian Forster, Matia Pizzoli, and Davide Scaramuzza. Svo: Fast semi-direct monocular visual odometry. In *2014 IEEE international conference on robotics and automation (ICRA)*, pages 15–22. IEEE, 2014.

[GBG12] Volker Grabe, Heinrich H Bülthoff, and Paolo Robuffo Giordano. Robust optical-flow based self-motion estimation for a quadrotor uav. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2153–2159. IEEE, 2012.

[Gib02] James J Gibson. A theory of direct visual perception. *Vision and Mind: selected readings in the philosophy of perception*, pages 77–90, 2002.

[GLT12] Dorian Gálvez-López and Juan D Tardos. Bags of binary words for fast place recognition in image sequences. *IEEE Transactions on Robotics*, 28(5):1188–1197, 2012.

[GLU12] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361. IEEE, 2012.

[GMAB17] Clément Godard, Oisin Mac Aodha, and Gabriel J Brostow. Unsupervised monocular depth estimation with left-right consistency. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 270–279, 2017.

[GMAFB19] Clément Godard, Oisin Mac Aodha, Michael Firman, and Gabriel J Brostow. Digging into self-supervised monocular depth estimation. In

Proceedings of the IEEE International Conference on Computer Vision, pages 3828–3838, 2019.

[GOGJ16] Ruben Gomez-Ojeda and Javier Gonzalez-Jimenez. Robust stereo visual odometry through a probabilistic combination of points and line segments. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2521–2526. IEEE, 2016.

[GWDC18] Xiang Gao, Rui Wang, Nikolaus Demmel, and Daniel Cremers. Ldso: Direct sparse odometry with loop closure. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2198–2204. IEEE, 2018.

[GZLY17] Xiang Gao, Tao Zhang, Y Liu, and Q Yan. 14 lectures on visual slam: From theory to practice. 2017.

[Hal83] John Hallam. Resolving observer motion by object tracking. In *Proceedings of the Eighth international joint conference on Artificial intelligence-Volume 2*, pages 792–798, 1983.

[HLFP13] Gim Hee Lee, Friedrich Faundorfer, and Marc Pollefeys. Motion estimation for self-driving cars with a generalized camera. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2746–2753, 2013.

[HMR09] Guoquan P Huang, Anastasios I Mourikis, and Stergios I Roumeliotis. A first-estimates jacobian ekf for improving slam consistency. In *Experimental Robotics*, pages 373–382. Springer, 2009.

[HS81] Berthold KP Horn and Brian G Schunck. Determining optical flow. In *Techniques and Applications of Image Understanding*, volume 281, pages 319–331. International Society for Optics and Photonics, 1981.

[HZ03] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[IA99] Michal Irani and P Anandan. About direct methods. In *International Workshop on Vision Algorithms*, pages 267–277. Springer, 1999.

[Jep] Allan Jepson. Optical flow estimation.

[JXL13] Yunliang Jiang, Yunxi Xu, and Yong Liu. Performance evaluation of feature detection and matching in stereo visual odometry. *Neurocomputing*, 120:380–390, 2013.

[KGL10] Bernd Kitt, Andreas Geiger, and Henning Lategahn. Visual odometry based on stereo image sequences with ransac-based outlier rejection scheme. In *2010 ieee intelligent vehicles symposium*, pages 486–492. IEEE, 2010.

[KGS$^+$11] Rainer Kümmerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige, and Wolfram Burgard. g 2 o: A general framework for graph optimization. In *2011 IEEE International Conference on Robotics and Automation*, pages 3607–3613. IEEE, 2011.

[KHK13] Sebastian Klose, Philipp Heise, and Alois Knoll. Efficient compositional approaches for real-time robust direct visual odometry from rgb-d data. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1100–1106. IEEE, 2013.

[KM07] Georg Klein and David Murray. Parallel tracking and mapping for small ar workspaces. In *2007 6th IEEE and ACM international symposium on mixed and augmented reality*, pages 225–234. IEEE, 2007.

[KM15] Kishore Reddy Konda and Roland Memisevic. Learning visual odometry with a convolutional network. In *VISAPP (1)*, pages 486–490, 2015.

[KSL17] Yevhen Kuznietsov, Jorg Stuckler, and Bastian Leibe. Semi-supervised deep learning for monocular depth map prediction. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6647–6655, 2017.

[KV18] Maria Klodt and Andrea Vedaldi. Supervising the new with the old: learning sfm from sfm. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 698–713, 2018.

[KYP$^+$17] Rahul Kottath, Durga Prasad Yalamandala, Shashi Poddar, Amol P Bhondekar, and Vinod Karar. Inertia constrained visual odometry for navigational applications. In *2017 Fourth International Conference on Image Information Processing (ICIIP)*, pages 1–4. IEEE, 2017.

[Lab06] Frédéric Labrosse. The visual compass: Performance and limitations of an appearance-based method. *Journal of Field Robotics*, 23(10):913–941, 2006.

[LAM$^+$19] Shing Yan Loo, Ali Jahani Amiri, Syamsiah Mashohor, Sai Hong Tang, and Hong Zhang. Cnn-svo: Improving the mapping in semi-direct visual odometry using single-image depth prediction. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 5218–5223. IEEE, 2019.

[LDIG11] Steven Lovegrove, Andrew J Davison, and Javier Ibanez-Guzmán. Accurate visual odometry from a rear parking camera. In *2011 IEEE Intelligent Vehicles Symposium (IV)*, pages 788–793. IEEE, 2011.

[LDMG$^+$17] Sifei Liu, Shalini De Mello, Jinwei Gu, Guangyu Zhong, Ming-Hsuan Yang, and Jan Kautz. Learning affinity via spatial propagation networks. In *Advances in Neural Information Processing Systems*, pages 1520–1530, 2017.

[LH81] H Christopher Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293(5828):133–135, 1981.

[LK$^+$81] Bruce D Lucas, Takeo Kanade, et al. An iterative image registration technique with an application to stereo vision. 1981.

[LLB$^+$15] Stefan Leutenegger, Simon Lynen, Michael Bosse, Roland Siegwart, and Paul Furgale. Keyframe-based visual–inertial odometry using nonlinear optimization. *The International Journal of Robotics Research*, 34(3):314–334, 2015.

[Low04] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[LSLR15] Fayao Liu, Chunhua Shen, Guosheng Lin, and Ian Reid. Learning depth from single monocular images using deep convolutional neural fields. *IEEE transactions on pattern analysis and machine intelligence*, 38(10):2024–2039, 2015.

[MAMT15] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE transactions on robotics*, 31(5):1147–1163, 2015.

[MCMD18] Travis Manderson, Ran Cheng, Dave Meger, and Gregory Dudek. Navigation in the service of enhanced pose estimation. In *International Symposium on Experimental Robotics (ISER)*, 2018.

[MHCD18] Travis Manderson, Juan Camilo Gamboa Higuera, Ran Cheng, and Gregory Dudek. Vision-based autonomous underwater swimming in dense coral for combined collision avoidance and target selection. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1885–1891. IEEE, 2018.

[MLD$^+$09] Etienne Mouragnon, Maxime Lhuillier, Michel Dhome, Fabien Dekeyser, and Patrick Sayd. Generic and real-time structure from motion using local bundle adjustment. *Image and Vision Computing*, 27(8):1178–1193, 2009.

[Mor80] Hans P Moravec. Obstacle avoidance and navigation in the real world by a seeing robot rover. Technical report, Stanford Univ Ca Dept of Computer Science, 1980.

[MS87] Larry Matthies and Stevena Shafer. Error modeling in stereo navigation. *IEEE Journal on Robotics and Automation*, 3(3):239–248, 1987.

[Mur62] FJ Murray. Perturbation theory and lie algebras. *Journal of Mathematical Physics*, 3(3):451–468, 1962.

[NKKUPVB18] Jogendra Nath Kundu, Phani Krishna Uppala, Anuj Pahuja, and R Venkatesh Babu. Adadepth: Unsupervised content congruent adaptation for depth estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2656–2665, 2018.

[OMSM01] Clark F Olson, Larry H Matthies, Marcel Schoppers, and Mark W Maimone. Stereo ego-motion improvements for robust rover navigation. In *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164)*, volume 2, pages 1099–1104. IEEE, 2001.

[PFC$^+$17] Taihú Pire, Thomas Fischer, Gastón Castro, Pablo De Cristóforis, Javier Civera, and Julio Jacobo Berlles. S-ptam: Stereo parallel tracking and mapping. *Robotics and Autonomous Systems*, 93:27–42, 2017.

[PKK18] Shashi Poddar, Rahul Kottath, and Vinod Karar. Evolution of visual odometry techniques. *arXiv preprint arXiv:1804.11142*, 2018.

[Pra80] K Prazdny. Egomotion and relative depth map from optical flow. *Biological cybernetics*, 36(2):87–102, 1980.

[RFB15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[RJB$^+$19] Anurag Ranjan, Varun Jampani, Lukas Balles, Kihwan Kim, Deqing Sun, Jonas Wulff, and Michael J Black. Competitive collaboration: Joint unsupervised learning of depth, camera motion, optical flow and motion segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12240–12249, 2019.

[SM16] Kumar Shaurya Shankar and Nathan Michael. Robust direct visual odometry using mutual information. In *2016 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 9–14. IEEE, 2016.

[SMR08] Geraldo Silveira, Ezio Malis, and Patrick Rives. An efficient direct approach to visual slam. *IEEE transactions on robotics*, 24(5):969–979, 2008.

[TC11] Tommi Tykkälä and Andrew I Comport. A dense structure model for image based stereo slam. In *2011 IEEE International Conference on Robotics and Automation*, pages 1758–1763. IEEE, 2011.

[TPD08] Jean-Philippe Tardif, Yanis Pavlidis, and Kostas Daniilidis. Monocular visual odometry in urban environments using an omnidirectional camera. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2531–2538. IEEE, 2008.

[TT18] Chengzhou Tang and Ping Tan. Ba-net: Dense bundle adjustment network. *arXiv preprint arXiv:1806.04807*, 2018.

[UESC16] Vladyslav Usenko, Jakob Engel, Jörg Stückler, and Daniel Cremers. Direct visual-inertial odometry with stereo cameras. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1885–1892. IEEE, 2016.

[Ull79] Shimon Ullman. The interpretation of structure from motion. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 203(1153):405–426, 1979.

[VSUC18] Lukas Von Stumberg, Vladyslav Usenko, and Daniel Cremers. Direct sparse visual-inertial odometry using dynamic marginalization. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2510–2517. IEEE, 2018.

[WBBP06] Joachim Weickert, Andrés Bruhn, Thomas Brox, and Nils Papenberg. A survey on variational optic flow methods for small displacements. In

Mathematical models for registration and applications to medical imaging, pages 103–136. Springer, 2006.

[WCR92] Juyang Weng, Paul Cohen, and Nicolas Rebibo. Motion and structure estimation from stereo image sequences. *ieee Transactions on Robotics and Automation*, 8(3):362–382, 1992.

[WCWT17] Sen Wang, Ronald Clark, Hongkai Wen, and Niki Trigoni. Deepvo: Towards end-to-end visual odometry with deep recurrent convolutional neural networks. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2043–2050. IEEE, 2017.

[WMBZL18a] Chaoyang Wang, Jose Miguel Buenaposada, Rui Zhu, and Simon Lucey. Learning depth from monocular videos using direct methods. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[WMBZL18b] Chaoyang Wang, José Miguel Buenaposada, Rui Zhu, and Simon Lucey. Learning depth from monocular videos using direct methods. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2022–2030, 2018.

[WP19] Xiaolong Wu and Cedric Pradalier. Robust semi-direct monocular visual odometry using edge and illumination-robust cost. *arXiv preprint arXiv:1909.11362*, 2019.

[WPF19] Rui Wang, Stephen M Pizer, and Jan-Michael Frahm. Recurrent neural network for (un-) supervised learning of monocular video visual odometry and depth. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5555–5564, 2019.

[WPKL07] Thomas P Webb, Richard J Prazenica, Andrew J Kurdila, and Rick Lind. Vision-based state estimation for autonomous micro air vehicles. *Journal of guidance, control, and dynamics*, 30(3):816–826, 2007.

[WSC17] Rui Wang, Martin Schworer, and Daniel Cremers. Stereo dso: Large-scale direct sparse visual odometry with stereo cameras. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3903–3911, 2017.

[WZL$^+$19] Xingkui Wei, Yinda Zhang, Zhuwen Li, Yanwei Fu, and Xiangyang Xue. Deepsfm: Structure from motion via deep bundle adjustment. *arXiv preprint arXiv:1912.09697*, 2019.

[YWGC18] Nan Yang, Rui Wang, Xiang Gao, and Daniel Cremers. Challenges in monocular visual odometry: Photometric calibration, motion bias, and rolling shutter effect. *IEEE Robotics and Automation Letters*, 3(4):2878–2885, 2018.

[YWSC18] Nan Yang, Rui Wang, Jorg Stuckler, and Daniel Cremers. Deep virtual stereo odometry: Leveraging deep depth prediction for monocular direct sparse odometry. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 817–833, 2018.

[ZBSL17] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G Lowe. Unsupervised learning of depth and ego-motion from video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1851–1858, 2017.

[ZGSW$^+$18] Huangying Zhan, Ravi Garg, Chamara Saroj Weerasekera, Kejie Li, Harsh Agarwal, and Ian Reid. Unsupervised learning of monocular depth estimation and visual odometry with deep feature reconstruction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 340–349, 2018.

[Zha06] Fuzhen Zhang. *The Schur complement and its applications*, volume 4. Springer Science & Business Media, 2006.

[ZTS19] Chaoqiang Zhao, Yang Tang, and Qiyu Sun. Deep direct visual odometry. *arXiv preprint arXiv:1912.05101*, 2019.

[ZWT03] Chao Zhou, Yucheng Wei, and Tieniu Tan. Mobile robot self-localization based on global visual appearance features. In *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)*, volume 1, pages 1271–1276. IEEE, 2003.

Acronyms

DVO	Direct Visual Odometry
CNN	Convolutional Neural Netowrk
SLAM	Simultaneous Localization and Mapping
KF	Key frame
GN	Gauss-Newton
LM	Levenberg-Marquardt
VO	Visual Odometry
BA	Bundle Adjustment
APE	Absolute Pose Error
RPE	Relative Pose Error
ATE	Absolute Trajectory Error
RMSE	Root Mean Square Error
SIFT	Scale-Invariant Feature Transform
BoW	Bag of Word